

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

TEXT EDITORS: The Baby Duck Syndrome

New Column on
Artificial Intelligence

6502 Killer Hacks

Efficient Hashing

Languages:

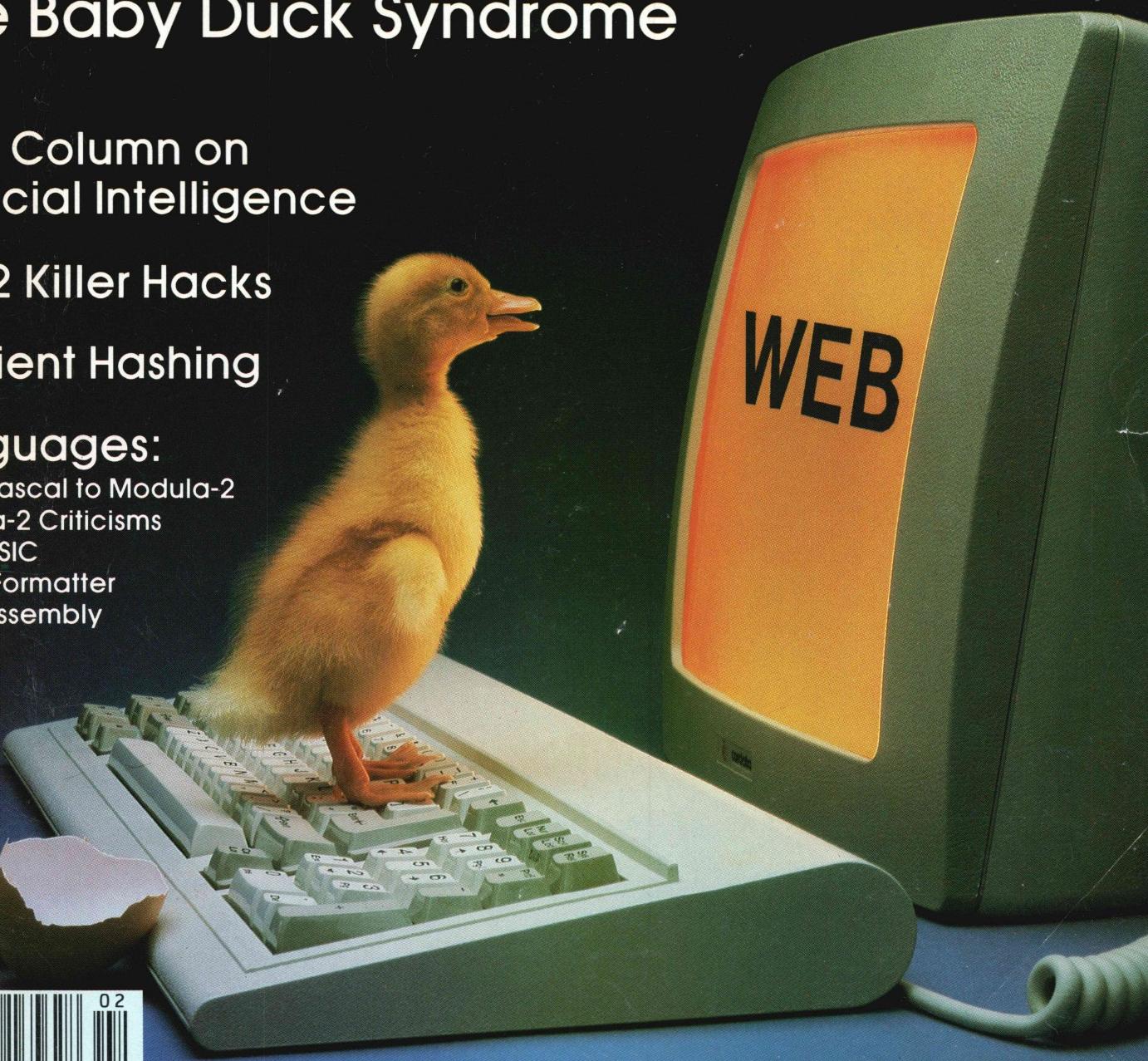
Turbo Pascal to Modula-2

Modula-2 Criticisms

True BASIC

C Text Formatter

6502 Assembly



Clipper is the fastest dBase III and dBase III Plus™ compiler available. Nothing else comes close. When performance counts, experts rely on Clipper for more speed, more power, and more creative freedom. You can, too. Call for details.

- Clipper compiled programs run 2 to 20 times faster.
- No royalties...no runtime fees.
- Source code security.
- User defined functions.
- Arrays.
- Simple menu commands.
- Context sensitive help can be included with programs.
- More fields; more memory variables.
- Call C and Assembly programs.
- Complete debugging facilities.
- Multiple file relationships.
- IBM PC, XT, AT, 3270 compatible™.
- Multi-user capability.

Clipper™

CLIPPER. THE dBASE COMPILER.
A WINNING PERFORMANCE EVERY TIME.



Nantucket™

Nantucket Corporation
5995 South Sepulveda Boulevard
Culver City, California 90230
(213) 390-7923

Outside California call toll-free:
1-800-251-8438

dBase, dBase III, and dBase III Plus are trademarks of Ashton-Tate, Inc.

IBM PC, XT, AT, and 3270 are trademarks of International Business Machines Corporation. Clipper and Nantucket are trademarks of Nantucket Corporation.

LOOK FOR CLIPPER™,
Autumn '86
IT MAKES NETWORKING EASY.

Circle no. 220 on reader service card.

TRANSFORM YOUR PC INTO A LISP MACHINE.



Gold Hill. The expert in AI on PCs.

*The Gold Hill 386 LISP System requires an IBM PC XT, AT or compatible. GCLISP 286 Developer Software requires an IBM PC AT or compatible.

© 1986 Gold Hill Computers, Inc. Gold Hill, Gold Hill 386 LISP System, Golden Common LISP, GCLISP, and Developer are trademarks of Gold Hill Computers, Inc. IBM PC, XT and AT are registered trademarks of International Business Machines Corp. HummingBoard is a trademark of A.I. Architects, Inc.

Circle no. 291 on reader service card.

Not long ago, a specialized LISP machine was your only choice for serious AI development and delivery.

But now you can get LISP machine performance out of that ordinary IBM PC * sitting on your desk. With the **Gold Hill 386 LISP System**.

You simply plug in the System's HummingBoard™—unique 386-based hardware designed specifically for the LISP environment. Then you add the System's Golden Common LISP 386 Developer software.

That's all you need to transform your PC into a LISP machine. You can develop and deliver AI applications on your PC. And create your own expert systems. You'll get the kind of LISP performance you thought was only possible in a system costing ten times as much.

And if you don't need the whole system, Gold Hill can still offer you AI solutions. You can get GCLISP 286 Developer software for your PC.* And GCLISP 386 Developer software will be available for leading manufacturers' 386-based PCs.

Tomorrow's AI development tools for PCs are available today from Gold Hill. We'll prove it to you—just call to get the latest Gabriel Performance Benchmarks. You'll be amazed to learn what your PC is capable of. Call toll-free:

1-800-242-LISP

In Mass.: (617) 492-2071
Gold Hill Computers, Inc.
163 Harvard St.,
Cambridge, MA 02139



FOR TURBO PASCAL



...one package stands out as the best support available for Turbo Pascal programmers: Blaise Computing's *Turbo Power Tools*. This definitive set of prewritten Pascal functions and procedures will make the life of any programmer—from the beginner to the hard-core professional—easier and more productive.

**ANOTHER
PLUS FROM
BLAISE
COMPUTING**

The best just got better! Turbo POWER TOOLS, acclaimed as the best programmer support package for Turbo Pascal, now has even more functions, more detailed documentation and more sample programs.

NO SECRETS

Turbo POWER TOOLS PLUS is crafted so that the source is efficient, readable and easy to modify. We don't keep secrets! We tell you exactly how windows are managed, how interrupt service routines can be written in Turbo Pascal, and how to write memory resident programs that can even access the disk. Maybe you've heard of some undocumented DOS features that resident programs use to weave their magic. Turbo POWER TOOLS PLUS documents these features and lets you make your own magic!

Here's just part of the PLUS in Turbo POWER TOOLS PLUS:

- ◆ **WINDOWS** that are stackable, removable, with optional borders and a cursor memory.
- ◆ **FAST DIRECT VIDEO ACCESS** for efficiency.
- ◆ **SCREEN HANDLING** including multiple monitor and EGA 43-line support.
- ◆ **POP-UP MENUS** which are flexible, efficient and easy to use, giving your applications that polished look.
- ◆ **INTERRUPT SERVICE ROUTINES** that can be written in Turbo Pascal without the need for assembly language or inline code.

Power Tools Plus™ Window Routines. Memory Resident Routines.

Routinely.

- ◆ **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple procedure calls, you can "schedule" a Turbo Pascal procedure to execute either when a "hot key" is pressed, or at a specified time.
- ◆ **PROGRAM CONTROL ROUTINES** allow you to run other programs from Turbo Pascal, and even execute DOS commands.
- ◆ **MEMORY MANAGEMENT** allows you to monitor, allocate and free DOS-controlled memory.
- ◆ **DIRECTORY AND FILE HANDLING** support to let you take advantage of the newer features of DOS including networking.
- ◆ **STRING** procedures allowing powerful translation and conversion capabilities.
- ◆ **FULL SOURCE CODE** for all included routines, sample programs and utilities.
- ◆ **DOCUMENTATION, TECHNICAL SUPPORT** and attention to detail that

have distinguished Blaise Computing over the years.

Turbo POWER TOOLS PLUS supports Turbo Pascal Version 2.0 and later and is just \$99.95.

Another quality product from Blaise Computing: **Turbo ASYNCH PLUS™**

A new package which provides the crucial core of hardware interrupt support needed to build applications that communicate. ASYNCH PLUS offers simultaneous buffered input and output to both COM ports at speeds up to 9600 baud. The XON/XOFF protocol is supported. Now it also includes the "XMODEM" file-transfer protocol and support for Hayes compatible modems.

The underlying functions of Turbo ASYNCH PLUS are carefully crafted in assembler for efficiency and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The high level function are all written in Turbo Pascal in the same style and format as Turbo POWER TOOLS PLUS. All source code is included for just \$99.95.

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

ORDER TOLL-FREE 800-227-8087

Calif. residents call (415) 540-5441

| | |
|--|--|
| YES, send me the PLUS I need! Enclosed is \$ _____ for | |
| <input type="checkbox"/> | Turbo POWER TOOLS PLUS <input type="checkbox"/> Turbo ASYNCH PLUS <input type="checkbox"/> |
| OTHER _____ (CA residents add 6 1/2% Sales Tax. All domestic orders add \$10.00 for Federal Express shipping.) | |
| Name: _____ | Phone: (_____) _____ |
| Shipping Address: _____ | State: _____ Zip: _____ |
| City: _____ | Exp. Date: _____ |
| VISA or MC #: _____ | |

ARTICLES

**Text editors: ►
whaddaya
want?**

**Hacking 6502 ►
assembly**

C text formatter ►

**MS-DOS ►
resources**

**Artificial ►
intelligence
today
Translating ►
Turbo Pascal to
Modula-2**

Knuth's WEB ►

**What's wrong ►
with Modula-2?
Memories ►**

TOOLS: Text Editors: In Matters of Taste... 16

by Levi Thomas and Nick Turner

The choice of a text editor is based partly on "hard" pragmatic requirements and partly on more subjective considerations, such as personal tastes and biases. In this month's cover story, *DDJ* talks to some programmers about the agony and the ecstasy of searching for the perfect text editor.

CODING: 6502 Hacks 24

by Mark S. Ackerman

Mark gives us a peek into the magician's hat. He describes some killer hacks designed to squeeze that last byte and/or machine cycle out of an assembly-language program.

ALGORITHMS: Hashing for High Performance 34**Searching**

by Edwin T. Floyd

Edwin explains, demonstrates, and evaluates four different hashing algorithms.

COLUMNS

C CHEST 90

by Allen Holub

The first in a series of installments on nr—Allen's version of the Unix formatting utility, nroff. This month he discusses symbol table maintenance with hashing functions, expression analysis, and a method for printing Roman numerals.

16-BIT SOFTWARE TOOLBOX 102

by Ray Duncan

A look at the MS-DOS STACK command, techniques for writing memory-resident programs, and books for MS-DOS programmers.

ARTIFICIAL INTELLIGENCE 108

by Ernest R. Tello

Our new columnist starts things off with some AI history, philosophy, and recommended reading.

STRUCTURED PROGRAMMING 124

by Namir Clement Shammas

Namir discusses program translation.

FORUM

PROGRAMMER'S
SERVICES**EDITORIAL 6**

by Michael Swaine

RUNNING LIGHT 8

by Nick Turner

ARCHIVES 8**LETTERS 10**

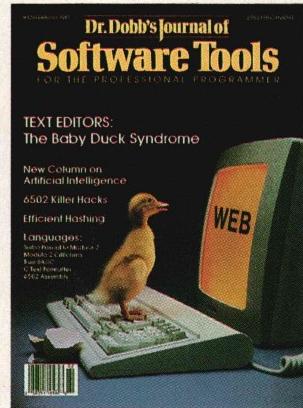
by you

VIEWPOINT 14

by Mike Suman

DDJ ON LINE 134**SWAINE'S FLAMES 152**

by Michael Swaine

**About the Cover**

Ethologists call it imprinting—a baby duck emerges from a shell and regards the first large object it sees as its mother. Is this why so many programmers still use WordStar?

This Issue

This month, we delve into some of the issues that must be considered when choosing—or designing—a text editor. We asked some programmers to tell us about their favorite and least favorite editing programs. Also, Allen Holub begins a series of columns on his own text processing program, nr.

Next Issue

In March, *DDJ* looks at data compression with a lead article that describes a recursive scheme for compressing image data and a comparative review of microcomputer archival programs.

Ctrace

```

253 main [variables] extern unsigned char
76   x[2][0]= .01;x[2][1]=.01;x[2][2]
77   x[3][0]=- .02;x[3][1]=.02;x[3][2]
78   printf("\n\nThe X matrix is");
79   for(n1=0;n1<a;n1++) {
80       for(n2=0;n2<a;n2++)
81           printf("\n\nx[%d][%d] is %f"
82   }
83   /* slash is at left hand end */
84   for(n1=0;n1<a;n1++) {
85       for(n2=0;n2<a;n2++) {
86           if(n2==n1)

```

| | 1 0402 3 |
|-----------------------|----------|
| todays.month | 9 |
| todays.day | 23 |
| todays.year | 86 |
| x[0][0] changed value | |
| y[0][0] changed value | |
| x1 = 2.00000e+00 | |
| x3 < 8.30000e+00 | |
| n1 > 9 | |
| n3 >= 33 | |

MATRIX INVERSION

Run number is 1

The X matrix is
x[0][0] is 1.000000
x[0][1] is 0.040000
x[0][2] is 0.030000
x[0][3] is 0.020000
x[1][0] is 0.020000_

| | 0x03fb |
|--------------|-------------|
| ptr | 9 |
| ptr->month | 23 |
| ptr->day | 86 |
| ptr->year | 'S' |
| ptr->name[0] | 'e' |
| ptr->name[1] | 'p' |
| ptr->name[2] | '\x00' |
| ptr->name[3] | 9.70865e-03 |
| f | 9.99909e-01 |
| t | 1.00000e+00 |
| x[0][0] | |

The perfect companion for MIX C has arrived. MIX C makes it easy to write C programs. Now Ctrace makes it easy to get them working. Introducing Ctrace, the exciting new C source debugger with animated trace.

FUN AND EASY

Ctrace makes it so easy to debug your C programs that you'll love doing it. You no longer have to mess with assembly language or hex addresses. Ctrace presents your program in a form that's instantly familiar. Your C source code is displayed just as you wrote it. All your variables are displayed just as you named them. And wait till you see your program in action. Ctrace brings it to life on the screen. You'll see your variable values changing as you watch your source code executing. Ctrace shows you how your program works, or why it doesn't work. After one session with Ctrace, you'll wonder how you ever programmed in C without it.

UNIQUE ANIMATED TRACE

Ctrace has a unique animated trace feature that shows you the flow of execution in vivid detail. Not just line by line, but statement by statement. It's like watching the bouncing ball as the cursor moves over your C source code, highlighting each statement as it executes. Press the space bar to execute one statement at a time, or press the return key and watch it go. It's exciting and educational. Who says learning has to be boring?

SIMPLE OPERATION

Ctrace is easy to operate too. Commands are executed with a single keystroke. Help screens are available if you forget a command. Pop up menus list command options. You simply position the cursor to the desired option and press the return key. Pop up messages alert you when anything important happens. To use Ctrace, simply compile your program with the trace option turned on. The executable program file is created as normal. Ctrace doesn't affect the size or the behavior of the program. You can execute your program with or without the help of Ctrace.

4 VIEWS AT ONCE

Ctrace maintains 6 windows of information: source, output, variables, watch, symbols, and memory. You can view as many as 4 windows at the same time. The source window (top left) shows your C program. The output window (bottom left) shows the screen output from your program. The variable window (bottom right) shows all the variable names and values. The watch window (top right) shows the variables that you select along with any conditions you've defined. The symbols window shows the addresses of variables and functions. The memory window shows any area of memory using data types that you select. Eight different screen layouts are available at the touch of a key. You can even define your own screen layouts.

COMPLETE PROGRAM CONTROL

Ctrace gives you complete control of your program. Execution options are single step, trace speed, and full speed. You can insert breakpoints on an unlimited number of statements. Execution is temporarily halted when a break point is hit. You can then

snoop around and see what your program has done to that point. You can even trace the flow of control backwards to see how your program got there. You can insert watch points on variable values. When the value of a variable satisfies the conditions you've defined, execution halts to let you examine your program. You can trace all functions or select just the ones you want to see.

THE RIGHT PRICE

If you could buy a debugger like Ctrace anywhere else you would expect to spend major bucks. Fortunately nobody else has a debugger like Ctrace. It's only available from MIX Software. And that's great news because you know our prices are right. Ctrace is an incredible value at only \$39.95. That's Right.



1132 Commerce Drive
Richardson, Tx. 75081
(214) 783-6001

Source window with profile count showing number of times each statement has executed. Pop up message indicates break point has been hit.

Source and variables windows shown side by side. Pop up message indicates that a watch point condition has been satisfied.

Source, variables, and memory window. Memory window lets you view any area of memory using any data type.

Change colors to suit yourself. Ctrace works with monochrome, color, Hercules, and EGA cards. Works on IBM compatibles and any computer with an IBM compatible BIOS.

C for yourself

THE C COMPILER

You can see that Ctrace is not your typical debugger. It's easy to understand and simple to operate. Likewise, MIX C is not your typical C compiler. It's small and fast. In fact it's the only full feature C compiler that can be operated comfortably on floppy disks. And as you would expect, MIX C is easy to use. It produces a complete program listing with all errors clearly identified and explained.

Although it's small, MIX C is not a subset. MIX C supports the full K&R standard, including the extensions that are often omitted in other C compilers. MIX C comes complete with a comprehensive 460 page book, a library of more than 175 functions, a blazingly fast linker, and tools for optimizing your programs for minimal space or maximum speed. All of this is yours for the incredibly low price of \$39.95. That's little more than the cost of most C books alone.

If you're just learning C, MIX C is your fastest, easiest, and cheapest way to master the language. If you've been frustrated by other C compilers, don't throw in the towel until you've tried ours. There's a world of difference. Our book includes a well written tutorial with lots of example programs. Our compiler includes the machine specific functions you need so you won't have to write them yourself. Compile and link operations take half as long with MIX C. That means you'll get your programs up and running twice as fast.

THE ASM UTILITY

Our ASM utility is available if you want to link assembly language functions to your C programs. It works with Microsoft's MASM or M80

assemblers. Macros make it easy! You can call assembly language functions just like C functions. You can even call C functions from assembly language. Lots of useful assembly language functions are included as examples. And the price is right at only \$10.

THE SPLIT-SCREEN EDITOR

Another great companion to the MIX C compiler is our split-screen editor. It makes writing programs even faster and easier. With the MIX Editor, you can compile, link, and execute your program at the touch of a key. Compiling is fast because the MIX C compiler reads the program directly from memory. Correcting errors is easy because the editor automatically positions the cursor to the first error in the program.

The MIX Editor works just like Micropro's WordStar. But through the magic of macros you can create your own custom version. You can map any key to any command. You can even define your own commands using the 100+ predefined commands. The split-screen feature is great for programming. You can edit two files at the same time and move text between files. It works great with any language. It has automatic line numbering for BASIC. It has auto-indent for structured languages like Pascal and C. It even has fill and justify for English. All these features and more are yours for the incredibly low price of \$29.95.

THE MIX C WORKS

The combination of Ctrace with MIX C makes C programming a real joy. MIX C provides the power of a compiler while Ctrace provides an execution environment that's more

elegant than an interpreter. Add the ASM utility and our versatile split-screen editor to the package and you've got a terrific C programming system. We call it the MIX C Works. What's great is that you can buy all four products for a fraction of the cost of other C compilers alone. Yes, buy all four and we'll give you a big \$29.95 discount off our already rock bottom prices. Only \$89.90 for the MIX C Works. Now that's a deal. That's Right.

MIX C WORKS

Only **\$89.90**

| Product | Price | Total |
|---|-----------|-------|
| Ctrace | (\$39.95) | \$ |
| C Compiler | (\$39.95) | \$ |
| ASM Utility | (\$10.00) | \$ |
| Split-Screen Editor | (\$29.95) | \$ |
| The MIX C Works | (\$89.90) | \$ |
| (includes all of above) | | |
| Subtotal | | \$ |
| Texas Residents Add 6.125% Sales Tax | | \$ |
| Add Shipping Charges | | \$ |
| In USA: add \$5 per order | | |
| In Canada: add \$10 per order | | |
| Overseas: add \$10 for editor | | |
| add \$20 for compiler | | |
| add \$30 for Works | | |
| Total of Your Order | | \$ |

TO ORDER CALL TOLL FREE:
1-800-523-9520

For technical support and for orders inside Texas please call (214) 783-6001

Or Contact one of our Worldwide Distributors direct in:

Canada: Saraguay 1-800-387-1288
France: Info/Tech 1-43-44-06-48
Australia: Techflow 047-586924
Switzerland: DMB CH-523-31817

System Requirements

Editor, C Compiler, & ASM Utility
MSDOS/PCDOS 2.0 or higher
128K Memory
1 Disk Drive
or CP/M 2.2 or higher (280)
55K Memory
1 Disk Drive (2 recommended)

System Requirements

Ctrace:
MSDOS/PCDOS 2.0 or higher
IBM compatible BIOS
256K Memory
1 Disk Drive

30 Day Money Back Guarantee Not Copy Protected

Please check method of payment

Check Money Order MC/VISA

Card # _____

Expiration Date _____

Please give name of computer _____

Please check operating system

MSDOS/PCDOS CP/M

Please check disk size

5 1/4" 3 1/2" 8"

Please check disk format if CP/M

SSDS SSDD DSDD

Your Name _____

Street _____

City _____

State _____ Zip _____

Telephone (_____) _____

Country _____

MIX
software

1132 Commerce Drive
Richardson, Tx 75081
(214) 783-6001

Ask about our volume discounts!
Dealer Inquiries Welcome

EDITORIAL

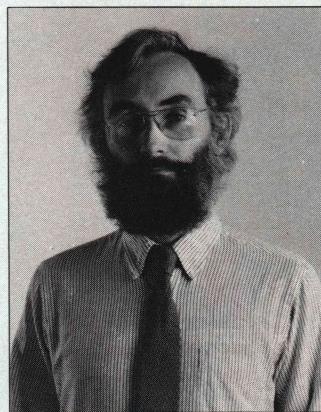
Our feature article this month deals with editors, and it is in some ways frustrating. It is certainly not one of those buyer's guide pieces that lead the bewildered consumer through the maze of creeping functions. We set out rather to examine the fundamental issues in editing, issues that both the user and the designer of a programmer's editor must face. What we found as we dug into the topic was that the more fundamental an issue was the more rabidly subjective it was likely to be—and arbitrary. The Baby Duck Syndrome is a force to be reckoned with. Again, frustrating.

Some recent developments, though not falling precisely in the domain of programmers' editors, suggest alternatives to traditional approaches to editing.

One of these is a product called Guide made by OWL International. The firm claims to have implemented Ted Nelson's vision of Hypertext in this Macintosh product. The claim itself is relatively uninteresting; Hypertext will be realized, is being realized, piecemeal, just like the grab bag of ideas that Alan Kay lumped together under the name Dynabook.

What is really interesting is that Guide has been explicitly designed to handle electronic documents. It will accept text files from conventional editors and will in turn produce flat text for conventional editors, but these tasks are outside its purpose. The document you structure with it lives in more than two dimensions.

OWL does not promote Guide as a tool for software development, but some of its features are interesting to look at in that context. Elements of the text can be suppressed or expressed at will, in a similar manner to the way an outline processor allows collapsing of detail but without the



Michael Swaine
Editor-in-Chief

constraint of the outline structure. A suppressed section is tied to a name, which it replaces when clicked on. This suggests automatic expansion of macros or pulling in a subroutine to examine its code. Guide also permits alternative text elements to be de-

fined and selected.

Another interesting development is the WEB documentation system, created by Donald Knuth and discussed in *The WEB System of Documentation* (Report Stan-CS-83-980) available from the Stanford University Computer Science Department. WEB actually implements some of the ideas described above in a tool specifically designed for writing structured and well-documented programs.

WEB is a high-level description language that produces Pascal code. WEB programs consist of short sections, each section comprising informal commentary, macro definitions, and Pascal code. The Pascal code can consist of a name and replacement text, in which case WEB will replace the name wherever it occurs with the replacement text (code).

One idea that both these developments suggest is the possibility of a program that would exist while under simultaneous development in two complementary forms: the working version and the working-on version. As Knuth puts it, "a WEB program is a Pascal program that has been cut up in pieces and rearranged into an order that is easier for a human being to understand. A Pascal program is a WEB program that has been rearranged into an order that is easier for a computer to understand."

Michael Swaine

Michael Swaine
editor-in-chief

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Editorial

Editor-in-Chief Michael Swaine

Editor Nick Turner

Managing Editor Vince Leone

Assistant Editors Sara Noah Ruddy

Levi Thomas

Technical Editor Allen Holub

Contributing Editors Ray Duncan

Michael Ham

Bela Lubkin

Namir Shammas

Ernest R. Tello

Copy Editor Rhoda Simmons

Production

Production Manager Bob Wynne

Art Director Michael Hollister

Assoc. Art Director Joe Sikoryak

Typesetter

Jean Aring

Cover Photographer Michael Carr

Circulation

Circulation Director Maureen Kaminski

Newsstand Sales Mgr. Stephanie Barber

Book Marketing Mgr. Jane Sharninghouse

Circulation Coordinator Kathleen Shay

Administration

Finance Director Kate Wheat

Business Manager Betty Trickett

Accounts Payable Supv. Mayda Lopez-Quintana

Accts. Receivable Supv. Laura Di Lazzaro

Advertising

Director Robert Horton (415) 366-3600

Account Managers

Michele Beaty (317) 875-8093

Lisa Boudreau (415) 366-3600

Gary George (404) 897-1923

Michael Wiener (415) 366-3600

Cynthia Zuck (718) 499-9333

Promotions/Srves.Mgr. Anna Kittleson

Advertising Coordinator Charles Shively

M&T Publishing Inc.

Chairman of the Board Ottmar Weber

Director C.F. von Quadrat

President and Publisher Laird Foshay

Associate Publisher Michael Swaine

Dr. Dobb's Journal of Software Tools (USPS 307690)

is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points.

Article Submissions: Send manuscripts and disk (with article and listings) to the Editor.

DDJ on CompuServe: Type GO DDJ

Address Correction Requested: Postmaster: Send Form 3579 to Dr. Dobb's Journal, P.O. Box 27809, San Diego, CA 92128. **ISSN 0888-3076**

Customer Service: For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

Subscriptions: \$29.97 per 1 year; \$56.97 for 2 years. Canada and Mexico add \$27 per year airmail or \$10 per year surface. All other countries add \$27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX: 620430 (WUI).

Entire contents copyright © 1987 by M&T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.

People's Computer Company

Dr. Dobb's Journal of Software Tools is published by M&T Publishing Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.



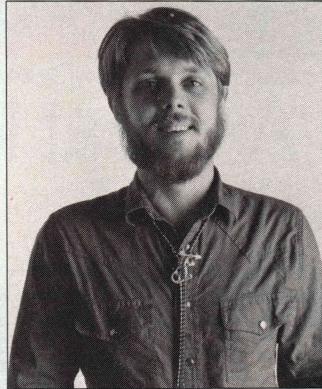
RUNNING LIGHT

This month we focus on a subject that is near and dear to the hearts of magazine editors and software authors alike: text editors, and what makes them good or bad. There have been many new developments in the field recently, and we wanted to get an idea of what you, as programmers, thought of them. Specifically, what is desirable in a program source code editor? Your opinions were interesting. Allen Holub also talks about editors in this issue: He begins a series of columns on nr, his version of the Unix nroff program.

Also in this issue, we introduce our newest columnist, Ernie Tello. Relatively sophisticated AI tools are finally becoming available, and Ernie talks about some of them in his first column. It's a welcome addition to *DDJ*, and I hope you'll send us a mountain of mail about it. Let us know what you think.

In this year's August issue, we'll take a look at tools for C programmers. We're particularly interested in articles that demonstrate the power and efficiency of really well-written C code. The ideal article would be between 1,000 and 3,000 words long and would include a listing between 100 and 400 lines long (see my listing advice below). Short file processors, keyboard filters, data compression techniques, and other utilities are all of interest, as well as math algorithms, string handlers, and other black box routines. If you've invented a tight new routine, let us know about it.

Here's more advice for authors, this time about program listings. *DDJ* is proud to be one of the few computer magazines that still regularly publishes source code listings. You want them, and we provide them. But list-



ings frequently turn into a headache at the layout stage, when we have to balance the cost of white space vs. the placement of ads and the need for the listings to be large enough to be readable. Another problem with listings is that frequently we have to do a lot of time-consuming editing to get them into a format that will print well on a laser printer. Sometimes editing of listings is necessary for another reason as well—the listing simply is not readable enough.

What can you do to help? It's really not all that hard. First, keep your listings as clean as you possibly can. That means no tab characters, for example. Instead, whenever possible use spaces to indent your code. Second, keep your listings on the narrow side—less than 60 columns if at all possible. If your comments go out beyond that point, I may shorten them or your article may be pulled from the issue because the listings won't fit. Readability is also important—if you are writing in C or Pascal (or some other structured language), use at least four columns for your indents. If it turns out that you can't keep your source code under 60 columns with four-column indents, then you have too many levels of nesting. The best thing to do in such cases is to remove the innermost levels and make them a separate routine. Some of you may be responding with outrage at this. You're the ones who put big banner headlines at the tops of your programs. That's fine for fanfold paper, but magazine space is often at a premium. So keep it short and sweet, please.

Nick Turner
editor

ARCHIVES

Program Editors

"It seems to me that a great obstacle to better programming is the lack of an editor that is as well adapted to its purpose as Visi-Calc is adapted to calculation in rows and columns."—*"Re-Thinking Program Editors," William B. Brodgen, *DDJ*, June 1981.*

Magazine Editors

"Planning each issue of *DDJ* is like playing a game of editorial bingo. We sit down at a long table with our game card and pieces of corn to lay on squares. Across the top of the card are the column labels—8080, 6502, 1802, 6800, etc. The horizontal rows are categorized into algorithm, languages, hardware, programming problems, and more. Now and then our pieces form a straight line and we all yell, 'Bingo!'"—*editorial, Marlin Ouvernon, *DDJ*, August 1981.*

Ten Years Ago in *DDJ*

"Talking dirty in a cryptogram is super except when you and your ten year old daughter (for real) decode it. It lost a lot of its humor while I tried explaining it."—*name withheld, letter to *DDJ*, February 1977.*

"I wish to offer a sincere apology for this. When I scanned the article, it occurred to me to check the sample for accuracy (but I didn't). However, it never occurred to me to check it for immature vulgarity...."—*Jim Warren, editorial response to the above, *DDJ*, February 1977.*

"For \$599, Ohio Scientific Instruments is selling a fully-assembled diskette drive including read/write electronics, manuals, mating connectors, system interface board (bare) and 6502/6800 operating system, delivery guaranteed to be less than 120 days. User supplied parts are conservatively estimated to cost an additional \$145. Eight to ten evenings of assembly time and testing are suggested."—*DDJ, February 1977.*

"R.E. Jef Raskin's 'Cardboard Computer Company'—I've had the same problem (money), so for a couple of bucks, I got some walnut grained contact paper which I used to cover cardboard, plywood, or aluminum chassis enclosures. If you take a little time to smooth the wrinkles and air bubbles, you will have the 'Classy Cardboard Computer Company.'... Oh yea, I really enjoy Jef's articles—wish he had written more about the S-100 bus earlier before I committed to a system that doesn't use said standard."—*W. B. Goldsmith, Jr., letter to *DDJ*, February 1977.*

DR. DOBB'S JOURNAL OF
COMPUTER
Calisthenics & Orthodontia
Running Light Without Overbite

A Challenge to Microsoft® C...

We challenge Microsoft C (Ver 4.0) to a C compiler duel to the finish, measuring compile, link, and execution times. If they win, we will stop advertising for two months.

by Roy Sherrill

If Microsoft C (Ver 4.0) can beat Optimum-C then we will stop advertising in all magazines for two full months and, win or lose, we will publish the results in its entirety. Even the Microsoft ads say "The Fastest C you've ever seen," so let the challenge begin.

Walter says Optimum-C is better

It all started when Walter Bright, the developer of Optimum-C, was explaining his new global optimizing C compiler and how it's code would be faster than Microsoft C (Ver. 4.0). Walter and I were frustrated because here we had a C compiler that would beat Microsoft C on 7 out of 10 benchmarks and also compile and link faster; yet our marketing consultant, Mark Astengo, told us that Microsoft C had a lock on the C compiler market and by 1990 they would probably have an 80% market share. Then Mark said, "Roy, if your C compiler is as fast as you say it is, why not challenge Microsoft C to a duel? If Microsoft wins, Datalight should stop advertising for two months and print the results of the test, win or lose." Well, I've always been one for a challenge. So here it is...

We only ask the following...

The benchmark suite will consist of the set of programs that Microsoft supplied to *Computer Language* for their February 1987 C compiler review issue. Microsoft will make available the programs to Datalight at least two weeks prior to the benchmarking. The benchmarking will be between Microsoft C 4.0 and Optimum-C. It will occur at a mutually agreed upon time and place. Interested individuals will be allowed to attend. The benchmarks will be compiled and run on a standard IBM PC-AT.

There will be two separate tests for each program: compile and link speed, and execution speed. For each test, a representative from each company will set up the compiler so that it performs at its best.

The benchmarks will be adjusted so that they take sufficiently long to run, that the tolerance involved in timing them is insignificant. The winner is determined by the compiler with the faster execution times for the majority of the benchmarks. We'd like an answer from Microsoft no later than April 1, 1987.

So what's a global optimizer?

A global optimizer looks at an entire function at once, analyzing and optimizing the whole function. A technique called data flow analysis is used by Optimum-C to gather information about each function. This enables your compute-bound programs to execute as much as 30% faster after global optimization. But, there is one catch...because the global optimizer ruthlessly searches for

ways to speed-up execution speed and minimize memory usage, it has relatively slow compile times. No need to worry, though, because you can merely turn the global optimizer off. In fact, you can select all, none, or partial of the following optimizations: constant propagation, copy propagation, dead assignment elimination, dead variable elimination, dead code elimination, do register optimizations, global common subexpression elimination, loop invariant removal, loop induction variables, optimize for space, optimize for time, very busy expressions.

Choose from five memory models

Speed your programs by selecting the memory model that best suits your application.

Memory Models

| Model | Code | Data |
|---------|-----------------------|------|
| Compact | 64k total code & data | |
| Small | 64k | 64k |
| Program | 1M | 64k |
| Data | 64k | 1M |
| Large | 1M | 1M |

Compiling, one step...

Now with the one step DLC program you can create .OBJ, .EXE and .COM files. Also, DLC can handle multiple files and run MASM on your assembly files.

Try Optimum-C risk free

Try Optimum-C for 30 days and if you are not 100% satisfied return it for a full refund. Also, for a limited time we are including a *free C tutorial which is a combination workbook and floppy disk to help lead you through the C language with tutorials, quizzes, and program exercises.

O.K. Microsoft, it's up to you. We've put two months of advertising on the line that says you can't beat Optimum-C to a real test. Your answer, please?

PRICES

Developer's Kit still only \$99
Optimum-C \$139
(includes library source code)

Add \$5 for shipping in US/\$15 outside US
COD (add \$2.50)

Not Copy Protected



ORDER TOLL-FREE TODAY!

1-800-221-6630

Microsoft and MS-DOS are registered trademarks of the Microsoft Corporation.

Circle no. 203 on reader service card.

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

COMPUTER LANGUAGE, February 1986

Optimum-C Version 3.0

- ♦ Full UNIX System 5 C language plus ANSI extensions
- ♦ Fast/tight code via powerful optimizations including common sub-expression elimination
- ♦ DLC one-step compile/link program
- ♦ Multiple memory model support
- ♦ UNIX compatible library with PC functions
- ♦ Compatible with DOS linker and assembler
- ♦ Third-party library support
- ♦ Automatic generation of .COM files
- ♦ Supports DOS pathnames, wild cards, and Input/Output redirection
- ♦ Compatible with Lattice C version 2.x
- ♦ Interrupt handling in C
- ♦ Debugger support
- ♦ ROMable code support/start-up source

MS-DOS® Support Features

- ♦ Mouse support
- ♦ Sound support
- ♦ Fast screen I/O
- ♦ Interrupt handler

MAKE Maintenance Utility

- ♦ Macro definition support
- ♦ MS-DOS internal commands
- ♦ Inference rule support
- ♦ TOUCH date manager

Tools in Source Code

- ♦ cat—UNIX style "type"
- ♦ diff—Text file differences
- ♦ fgrep—fast text search
- ♦ pr—Page printer
- ♦ pwd—Print working directory
- ♦ wc—Word count

Datalight

Box 82441
Kenmore, Washington 98028
(206) 367-1803

*Limited offer available exclusively to readers who purchase directly from Datalight.

LETTERS



80386

Dear DDJ,

I have finished reading "Programming on the 80386" (October 1986), and I am both excited and troubled by all of the changes (enhancements) made to the basic 8086.

It takes several years for the community of software professionals to master a machine—to make that computer do more than it is generally believed it can do. There is no question that the new operating systems and application programs for the 80386 are going to be very sophisticated (or should be). Just using the machine's instructions the way they are intended to be used scares and attracts me at the same time.

But it takes a long time to master a computer at the level of a good assembly-language programmer. It takes less time for higher-level languages, but a state-of-the-art application writer still needs to know how to take advantage of the services offered by the operating system. This knowledge is best gained by the shared experience of many users.

Only after the experience of the whole community has reached a certain level (which can take years) does it become general knowledge what not to do on a particular machine. This information is so necessary and so important that all major bulletin-board services have subgroups for all the major types of computers that serve as a forum for their user communities.

There is already a shortage

of software professionals who program on dinosaur computers in near-dead languages. And the new languages (Ada, for example) will require several years of hands-on experience before there are many people who know how to use them.

The learning curve for software professionals lags behind the advancement of technology, for both hardware and software, by several years. If students use a particular machine that is state of the art when they are sophomores, by the time they are seniors, their knowledge will be only somewhat useful. If dynamic programmers stay at one company doing one type of programming on one type of machine with one type of operating system for more than just a few years, their knowledge will become obsolete unless they make a conscious effort to be aware of industry changes.

This problem—the too-rapid advancement of technology—is nothing new. My question to you is, How can the community of software professionals keep up with all the advances in hardware? It's difficult enough just to read a fine magazine such as *DDJ* every month. But to have

true knowledge of how to program a computer well is something that you have to learn by doing. We learn from our mistakes.

I feel that the solution is a series of hands-on courses for professionals on advanced topics in hardware and software. I know computer vendors offer courses, but usually they are priced for the budgets of data processing departments instead of individuals and they rarely explain techniques used by their competitors.

I'm sure there's a market for this kind of education. Just computer consultants and students alone would fill enough seats to make it worthwhile. And with all those hot shots under one roof, the class discussions would be interesting (if not more informative than the lessons).

Robert Rouse
479 Northlake Dr., #108
San Jose, CA 95117

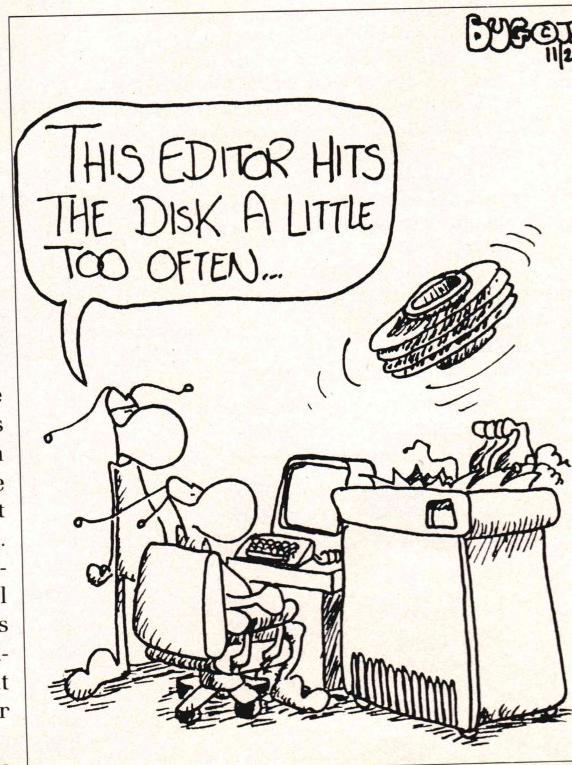
Dear DDJ,
I couldn't let Table 1 in Ross Nelson's article on the 80386 in the October 1986 issue go by without a response. Although I am no fan of the 8088, he has made it look worse than it really is. No self-respecting compiler would generate the code he listed in Table 1. I suggest my Table 1, page 12, as a replacement for his.

Tom Pennello
MetaWare Inc.
903 Pacific St., Ste. 201
Santa Cruz, CA 95060

Drowned in C

Dear DDJ,
This is in response to letters in October 1986 responding to the June 1986 Viewpoint, "What's Wrong with C." Though I found these opinions interesting, I think they have all missed what's really wrong with C.

Having known Pascal for years, I began learning C less than a year ago. In the time I've known Pascal, I've cursed BASIC (the favorite programming language of those who don't know any other language) because it allows pro-





When Will It Dawn On You?

XTC® is a PC programmer's dream come true. Just ask the thousands of programmers who've already awakened to the extraordinary features of this world class editor, designed for the IBM PC and true compatibles.

Features like XTC's interpretive macro language that lets you code macros on the fly. And fine-grained multitasking so you can run macros in the background while you continue editing.

Of course, XTC also lets you code macros in a high level language. And with the macro compiler you can write macros to check syntax in real time and translate source code from one language to another.

What's more, DOS compilers, linkers and utilities will run inside XTC so you don't have to leave the editor to compile and test run your programs.

If that's not enough, XTC has multiple large windows, 20 text buffers, and a host of other powerful features no other editor can offer.

In brief, you'll find little to compare with the power, flexibility, and advanced features that make XTC outshine the competition every time.

So if you're still dreaming about the ultimate editor, wake up. It could be right before your eyes.

XTC. From WENDIN. Only \$99.

ORDER HOTLINE

(509) 624-8088
(MON.-FRI., 8-5 PACIFIC TIME)



WENDIN®

327 E. PACIFIC
SPOKANE, WA 99202

© Copyright 1986 WENDIN, Inc.

The people who make quality
software tools affordable.

Ask about our other products
for the IBM PC and true
compatibles.

PCVMS™

Multitasking, multiuser version of
DEC's powerful VAX/VMS operating
system. Runs most MS-DOS
programs.

PCNX™

True multitasking, multiuser
operating system similar to AT&T's
popular UNIX® operating system.

OPERATING SYSTEM TOOLBOX™

Complete software construction set
that lets you build your own
multitasking, multiuser operating
systems.

All products priced at \$99
with source code included.

DEALER INQUIRIES WELCOME

Foreign orders inquire about shipping.
Domestic orders add \$5.00/1st item, \$1.00 each
additional item for shipping, handling, and
insurance. We accept Visa/MC, American
Express, COD, and Bank Drafts drawn on U.S.
Banks.

Washington residents add 7.8% sales tax.

MS is a trademark of Microsoft, PC-DOS is a
trademark of IBM. UNIX is a trademark of AT&T.
VAX/VMS is a registered trademark of Digital
Equipment Corporation.

**WENDIN and XTC are registered trademarks of
WENDIN, Inc. PCNX, PCVMS, Operating System
Toolbox, and Personal Operating System are
trademarks of WENDIN, Inc.**

grammers to make errors that are impossible in most other languages. (When typing code into interpreted BASICs, you can easily mistype a line number, causing that line to be somewhere else in the program and possibly erasing a previous line with the same number.) While learning C, I discovered errors that are unique to it, also. Some of these are related to the implementation I've been using (MS-DOS Lattice), but others are a result of the C language definition.

Many of the errors I've made relate to the almost total lack of type checking in C. The following code will demonstrate:

```
int i, ar[10];
for (i=0; i++; i < 10)
    ar[i] = 0;
```

If you don't see the problem with it immediately, you might spend a few hours looking at other parts of the program, as I did. The *for* statement first sets *i* to 0, then executes the second statement (*i++*) and exits from the loop if its result is 0. The statement *i++* returns a 1, and so the loop continues. The body of the *for* is executed, setting *ar[1]* to 0. Then the statement *i < 10* is executed (which I obviously wanted to be my terminating condition), and its value is not used. The loop continues while *i* is not 0, which on a compiler that uses 16-bit integers is 65,536 times. This happened because I accidentally swapped parts of the *for* statement, and because Boolean values are just integers in C, the compiler blindly accepted and compiled it. Another problem is that the integer array is indexed with this value and, because of a lack of array bounds checking, initializes 131,072 bytes to 0—much more than was desired. Strangely enough, my program did not crash, but a rather odd thing happened—the output to the screen was “buffered,” and nothing would print until the internal “buffer” became filled with 256 characters.

Another type checking problem has to do with function parameters. The compiler I use doesn't check that the parameter types in a function definition match those in a function call or even whether the number of

parameters is the same. I was under the impression that this was excusable only in older languages, such as FORTRAN, and not in more modern, structured languages. The existence of separate utilities such as lint to check such things tells me that this type of error checking was simply left out of compiler definitions.

At least some type checking is done in C, though, and it's with the return values of functions. I was getting warning messages with functions that didn't return any value. My initial kludge fix was to add the statement *return (0)*; at the end of each function, which while satisfying the compiler, I presume generated at least one extra instruction to set the return value, which wasn't even used in the function call. But this still didn't fix the warning message for function *main()*. Then I discovered that the *void* keyword, a new ANSI addition to C, is used to define a function that returns no value. But this still didn't quite work for me until I remembered that a function of type other than *integer* must also be declared explicitly, either in the calling function or as a global. The last time I remember something being default unless declared otherwise was in FORTRAN, in which variables beginning with the letters *I* through *N* were default *integer* and others were default *real*. As I became more proficient in FORTRAN (shortly after learning Pascal), I started defining all the variables in my FORTRAN programs explicitly and ignored the defaults. This also improved the clarity of my variable names. Both Pascal and FORTRAN allow functions that don't return a value, with the keywords *Procedure* and *SUBROUTINE* as part of the original language definitions. And Pascal, by not having defaults,

doesn't make programmers learn (and possibly forget) extra rules and their exceptions.

One more source of possible errors in C involves macro preprocessing, specifically macros that look like functions. This (and other side effects—C seems to have more possible side effects than any other language) is actually documented in compiler manuals. A sample statement looks like:

```
c = toupper (getchar());
```

in which *toupper* is defined as a macro. When expanded, this line actually has more than one call to *getchar()*, resulting possibly in several characters being read when only one was wanted. Another aspect of this is that *toupper()* could be a macro or a function (both are included with Lattice), depending on what header files are included. If it's a function, the preceding code will work just fine. It's not hard to imagine someone porting that line of code and, on seeing it, recognizing *toupper()* as a macro and including a header that defines the macro *toupper()* and thus breaking something that didn't need fixing.

I see the biggest problem with C is its growing popularity and thus the growing availability of C compilers, function libraries, and utilities, which take efforts away from support of other structured languages such as Pascal or even something better—the development of yet another new language, one that all of us could hope would not have such problems.

Ben Bradley

Telecorp Systems Inc.

5825-A Peachtree Corners E
Norcross, GA 30092

(continued on page 130)

| 8086 | 286 | 386 |
|----------------|----------------|-------------------|
| MOV BX,I | MOV BX,I | MOV EAX,I |
| SHL BX,1 | SHL BX,2 | |
| SHL BX,1 | | |
| FLD FOO+12[BX] | FLD FOO+12[BX] | FLD FOO+12[EAX*4] |
| FSQRT | FSQRT | FSQRT |

Table 1: Suggested replacements for Ross Nelson's implementation of SQRT (FOO[I+3])



Some Of The Most Famous Faces In Software Use Our C Functions

Our famous customers are a little camera shy. It's not that they are embarrassed by being Essential C Utility Library users. They just don't want us shouting their names from the roof tops.

The prestige of our users is not the primary reason to buy the Essential C Utility Library. The increased speed, features, and size efficiency of our products are the factors that demand their use. Our library contains *over 400 functions*, all designed with elegance in mind.

However, should curiosity get the best of you, call us at 201-762-6965 and we'll drop a few highly impressive names on you.



Behind every great program is a great library.

What's a Library Without a Librarian?

Our library comes complete with a sophisticated source code librarian. Now you can maintain current versions and conserve disk space. We want your development work to go as smoothly as possible.

No Royalties, 30 Day Guarantee

If within 30 days you don't find our library totally satisfactory, bag the whole thing and receive a complete refund. There are no royalties associated with the library.

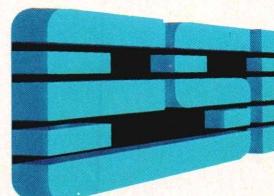
Functions At A Glance

- Fastest screen output available.
 - Save/Restore color screens in 1/10 sec.
 - Pop-up block cursor menus
 - Save/Restore windows to disk or memory
 - 50 functions for business graphics
 - dozens of string formats
 - time and date arithmetic
 - julian and day-of-week
 - Ctrl-Break key trapping
 - Field oriented data entry
 - Stuff keyboard buffer
 - 18 Mouse control functions
 - Execute programs and batch files
 - Disk error trapping
 - Determine space available
 - 40 functions to process characters and words
 - Insert, delete, extract, index, translate
 - Tested, easy-to-follow examples
 - Demo programs with source code
 - All source code included
- Documentation:**
Thorough, comprehensive. 260 pages
- Compatible C Compilers:**
Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, DeSmet, and Wizard

\$185.00

Do Your Homework

The library you buy can influence the rest of your programming life. We encourage you to do some checking before making a decision. When you've done your homework, you'll choose Essential. Call our support staff of experienced C programmers and find out before you buy how things will be after your check clears.



To order or for support
call: 201-762-6965

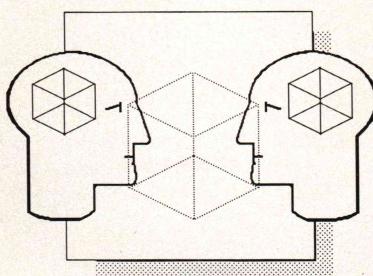
For foreign orders contact:

England: Gray Matter Tel. (0364) 53499
Japan: Lifeboat Inc. of Japan Tel: 293 4711
West Germany: Omnitex Tel. 07623-61820

Essential Software, Inc.
P.O. Box 1003, Maplewood, New Jersey 07040

Circle no. 138 on reader service card.

VIEWPOINT



What's Wrong with High-Level Languages?

The code accompanying Brian R. Anderson's article "A 68000 Cross Assembler" (DDJ, April and May 1986) provides an opportunity to criticize higher-order languages in general and Modula-2 in particular. It is not my intention to criticize Anderson. I enjoyed his article, and his Modula-2 code constitutes, at the very least, an impressive effort. But the code itself, precisely consistent with the teachings of our leading authorities, reveals deficiencies that cannot be allowed to pass.

The code in Code Example 1, right, is a fragment from Anderson's Listing Sixteen. I have left out comments and some details not relevant to my

by Mike Suman

points. The last seven lines of code are followed by 116 similar entries, ending with

```
INC (i);
WITH Table68k[i] DO
  Mnemonic := "UNLK";
  Op := {14, 11, 10, 9, 6, 4, 3};
  AddrModeA := ModeA {Ry02};
  AddrModeB := ModeB {};
END;
```

There then follows code to write out the array and finally the line

Mike Suman, 332 Sturtevant Dr., Sierra Madre, CA 91024. Mike started working with computers in 1949. He has been the director of research and development at an aerospace firm and is currently writing a book about computers.

END InitOperationCodes.

First, let me comment on a minor matter of style over which some leading programmers have poured major words. In this program fragment, the constants *FIRST* and *LAST* have been clearly set out in the beginning. As the matter is taught, this is supposed to make the program clearer, more immediately evident, and easier to modify.

But the only place in which *FIRST* and *LAST* are used is in the phrase *Table68K [FIRST..LAST] OF TableRecord*. This is surely *not* more immediately evident than saying *Table68K [1..118] OF TableRecord*. And no one can maintain that it is really easier, or safer, to change values in an early definition than it is to change them in the only place in which they are used later, when it is obvious what the effects of the change are going to be.

As a general principle, common sense and literary history both argue

that it seldom simplifies or clarifies complex matters to introduce new names for things that are already named, as are numbers, or to separate numerics from the phrases that reference them. You don't make "2 plus 2 equals 4" clearer by saying "x is 2 and y is 4 and x plus x equals y." The issue is small, but the point is large: We do not tolerate this kind of turgid excess in writing; why then do we force it in programming?

Having begun on a small point, let me move to a larger one. Imagine that you were handed a list of groceries that had been "carefully arranged for clarity":

| | |
|------------|-----------|
| Item | potatoes |
| Quantity | 3.2 lbs |
| Unit cost | \$0.65/lb |
| Total cost | \$2.08 |
| Item | oranges |
| Quantity | 5 lbs |
| Unit cost | \$0.88/lb |

(continued on page 132)

```
MODULE InitOperationCodes;
CONST
  FIRST = 1;
  LAST = 118;
TYPE
  ModeTypeA = (RegMem3, Ry02, Rx911, ..., OpM37);
  ModeTypeB = (Bit811, Size67, Size6, ..., EA611);
  ModeA = SET OF ModeTypeA;
  ModeB = SET OF ModeTypeB;
  TableRecord = RECORD
    Mnemonic : Token;
    Op : BITSET;
    AddrModeA : ModeA;
    AddrModeB : ModeB;
  END;
VAR
  Table68K [FIRST..LAST] OF TableRecord;
  i : CARDINAL;
BEGIN
  i := 1;
  WITH Table68K[i] DO
    Mnemonic := "ABCD";
    Op := {15, 14, 8};
    AddrModeA := ModeA{Rx911, RegMem3, Ry02};
    AddrModeB := ModeB{ };
  END;
  INC (i);
  WITH Table68K[i] DO
    Mnemonic := "ADD";
    Op := {15, 14, 12};
    AddrModeA := ModeA{OpM68D};
    AddrModeB := ModeB{OpEA05Y};
  END;
```

Code Example 1: Fragment of code from Listing Sixteen, DDJ, May 1986

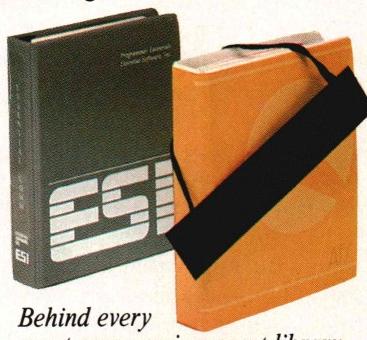


You'd Be Surprised Who Uses Essential Graphics Functions

And some of our well-known customers would be just as surprised if they saw themselves in this ad. We don't splash their names all over the place as a matter of professional courtesy.

Let's face it. Just because someone else uses a product is not reason enough to buy it. The clincher is that our programs run a *documented 40% faster* than the closest competitor. To complete the picture, our code is up to *75% smaller* due to efficient coding and the granularity of functions.

However, should curiosity get the best of you, call us at 201-762-6965 and we'll drop a few highly impressive names on you.



Behind every great program is a great library.

Draw Your Own Conclusions

When you're responsible for a project that includes advanced graphics, "graphics windowing," or character font manipulation, Essential Graphics is the clear choice. We've taken the grind out of graphics programming and replaced it with speed and versatility.

No Royalties, 30 Day Guarantee

We believe that selling you a programming tool does not make us your co-authors. So we don't charge any royalties or run time fees. If within 30 days you don't find our library satisfactory, dump the whole thing and receive a complete refund.

Functions At A Glance

Features:

- Fastest functions available
- Dots, Lines, Circles, Arcs, Pies, Bars
- Manipulate character fonts
- Move blocks, do animation
- User definable patterns
- Seed filling in a boundary
- Clipping on screen coordinates

Devices Supported:

- IBM, Epson, Oki printers
- HP Plotters, HP Laser Jet
- Microsoft, Logitech Mice

Graphics Adapters:

- IBM Color Graphics
- IBM Enhanced Graphics
- Hercules Graphics
- AT&T, Olivetti Graphics
- Tecmar Graphics Master
- Others (Call)

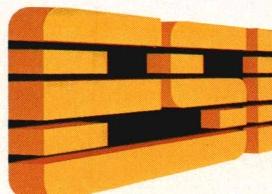
Compiler Compatibility:

- Microsoft, C, Fortran, Pascal
- Lattice C, Aztec C
- Computer Innovations C86, DeSmet C
- Wizard C, Mark Williams

\$250.00

Do Your Homework

The library you buy can influence the rest of your programming life. We encourage you to do some checking before making a decision. When you've done your homework, you'll choose Essential. Call our support staff of experienced C programmers and find out before you buy how things will be after your check clears.



To order or for support
call: 201-762-6965

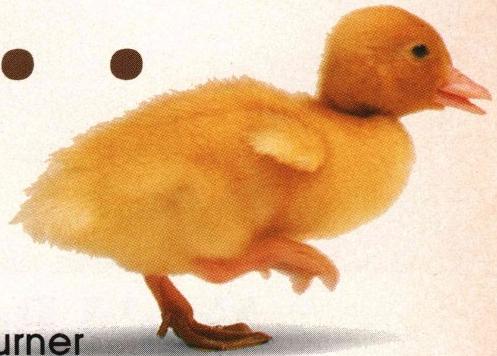
For foreign orders contact:

England: Gray Matter Tel. (0364) 53499
Japan: Lifeboat Inc. of Japan Tel: 293 4711
West Germany: Omnitex Tel. 07623-61820

Essential Software, Inc.

P.O. Box 1003, Maplewood, New Jersey 07040

Text Editors: In Matters of Taste . . .



by Levi Thomas and Nick Turner

"Editors? You wanna talk editors? How's about something trivial like life after death, religion, or politics?"—Don Watkins, sysop on Compu-Serve's IBM NET.

History

"The first interactive editor I ever used was Expensive Typewriter on the PDP-1 at MIT (yeah, the one we played Spacewar on)."—Dennis Brothers, author of Mactep and MicroPhone

Back in the good old days, people seldom had much choice when it came to text editors. (How many of you remember TECO or Expensive Typewriter? How about keypunches and Hollerith cards?) There wasn't a lot you could do with a simple keyboard and a hard-copy printer. Editing was almost exclusively line-oriented, and the only way to get an idea of what the text actually looked like was to list a section of the file explicitly, usually by specifying a range of line numbers. Frequently the editing commands were cryptic and hard to learn, such as *34LSj10U20D\$* (an actual command string from an early line editor). Things have certainly changed since those days. CRT screens and mice have been invented, and all sorts of user interface discoveries have been made. You might think that by now someone would have invented an editor so nearly perfect that it would set a standard imitated by all the rest—but nope.

Although the incomprehensible command strings of yesterday are gone from the new editors, replaced in most cases by sensible logical structures, the old editors live on. People still cling to WordStar, with all its hard-to-

'Which is better? It's a matter of personal bias—of taste.'

memorize control codes, despite the barrage of attacks from other camps. In fact, programmers who admit to still using WordStar often get the same kind of reception as programmers who announce that they prefer BASIC to Pascal.

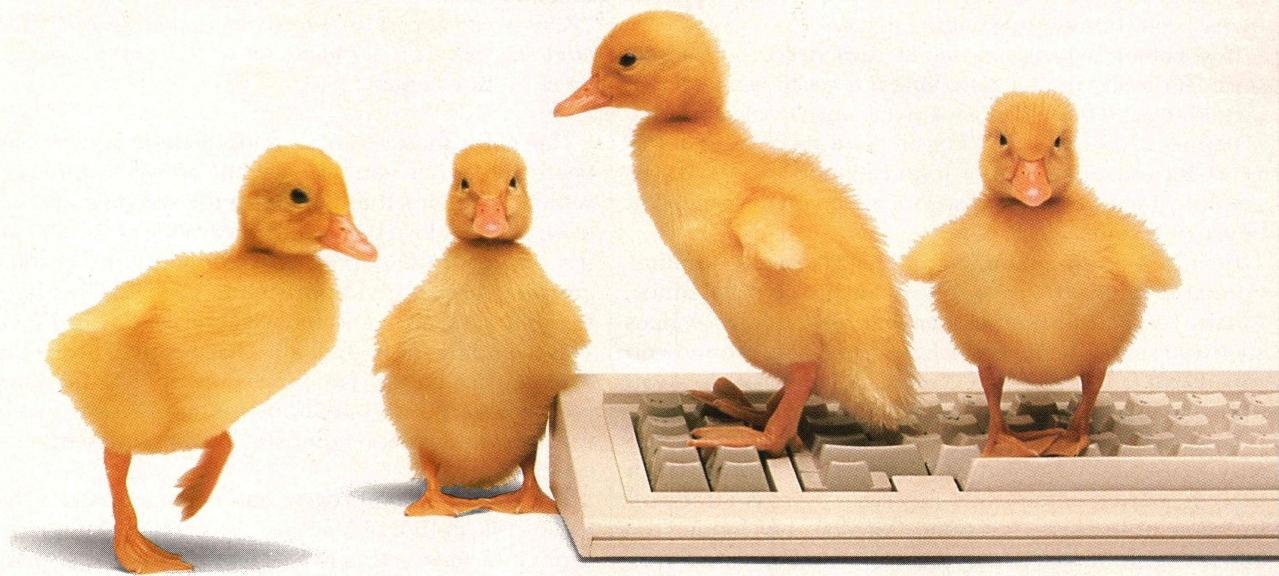
So, why do so many of the older, presumably outdated, editors still live? Well, the "if it's not broken, don't fix it" philosophy accounts for it somewhat—some folks are quite happy with their "outdated" editors and ignore the "if it's newer it must be better" attitude that is so prevalent in this industry. But more often imprinting seems to have a lot to do with it.

Old Workhorses and Baby Ducks

"I still use WordStar in nondocument mode when editing programs. OK, so kill me."—Ray Duncan, DDJ columnist

Like ducklings that adopt the first moving object they see as a mother, programmers often adopt the first editor they learn as the model of what an editor is and should be. Once you've learned an editor, once it is "burned into your brain," it may not be worth the effort to learn another (or, more to the point, to unlearn the first editor) no matter how comparatively easy it is or how many "neat" features it has. This learning process is especially frustrating because "everything you know is wrong"—you are accustomed to being in control of the process, of not having to think about the steps that stand between you and what you want done. When you learn a different editor, that transparent process becomes pitifully opaque. What a gumption trap! You see a similar occurrence when writers try to learn to use a word processor. Even though the rewards are enormous, the first few hours or days are

Levi Thomas and Nick Turner, 501 Galveston Dr., Redwood City, CA 94063. Levi and Nick are editors for DDJ.



hell. Your concentration is constantly interrupted by the mundane mechanics of word processing, and you feel as though there is suddenly a great barrier between you and your work—a barrier you didn't notice when you used the typewriter. So once you become comfortable with a word processor (or an editor), you will be pretty reluctant to start over again.

Buddy Can You Paradigm?

There seems to be far more disagreement than agreement when it comes to text editors. The dissent runs deep, and it's not just a matter of taste (although that certainly enters into it as well). Not only are there different ideals for different tasks, but also individual style has a major impact. Sometimes it seems there are as many schools of thought as there are programmers. And programmers do tend to be adamant about their likes and dislikes in this matter.

What Do We Agree On?

There are some features we all seem to agree are vitally important in a "good" editor. These are the factors that apply to all editing situations, on all systems . . . and they are few.

Speed

"An editor must be fast. If it's not blindingly fast in screen updates (or at least as fast as possible), I will probably put my fist through the CRT screen after the 1,793th line of code at 1 A.M. in the morning."—Darryl Okahata, programmer

No one likes to wait for anything, especially when staring at a computer screen. If an editor pauses for more than a fraction of a second for any reason, programmers often perceive it as a serious flaw.

Today's editors use advanced techniques such as the Boyer-Moore string search, hashing tables, and RAM

caches to speed up performance. More RAM has also meant that entire files can often be held in memory, whereas before they had to be paged or sometimes could not be edited at all.

Features

Many of today's editors are packed with sophisticated features—some of which are seldom really needed. Although it is possible for an editor to be so feature-laden that it becomes cumbersome and difficult to learn, programmers do agree that that an editor must have a basic set of powerful functions. Just what that set of functions might include is a source of much debate.

User Interface

"An editor is something you really curl up with. If two editors have the same features, you're going to go with the one that 'feels' right. It's like professional musicians are about their instruments . . . some choose a Gibson Hummingbird while others may prefer a Martin. Which is better? It's a matter of personal bias—of taste."—Bob Wallace, author of PC Write

An important and highly subjective issue is ease of use. The best editor is transparent—its use becomes so natural that you forget it's there. Ease of use is a difficult thing to measure, but some traits are worth examining. For example, editors that are "modeless"—that is, the program rather than the human using it keeps track of the mode—have generally become accepted as superior to ones that require humans to keep track. Of course, there is still a lot of disagreement as to how this is best implemented.

But almost any editor is easy to use once you've mastered its syntax and commands. Just what kind of learning curve you are willing to tackle is the big question. Often the sweat and frustration of learning an editor with complicated arcane commands is the price of gaining a

TEXT EDITORS

(continued from page 17)

high degree of control over the editing process—a price many programmers are willing to pay.

Roy LeBan, a programmer at Ann Arbor Softworks, says: "It doesn't matter how long it takes to learn an editor, as long as it does what I want it to once I've learned it."

Dennis Brothers says: "My all time favorite editor is TECO, for emotional rather than rational reasons. It's an absolute bitch to learn to use, but once you've learned it, boy, can you make it dance!"

Dennis Allison, cofounder of the People's Computer Company (and *DDJ*), says this about his favorite editor, Emacs: "It's hard to remember that meta-Control-X does such and such, but once you learn your way around you can do all sorts of things. You can, for instance, swap every other word with a single editing command."

Fortunately, a variety of editors available today are easy to use and require a lot less time and effort to learn than the earlier editors. In many cases there is a trade-off in terms of features, but often this does not hinder the editor's usefulness for most jobs. These editors are a welcome alternative for most programmers, particularly those who take umbrage about complicated, counter-intuitive command strings.

Alex Pournelle of Workman and Associates puts it succinctly: "I might adopt Unix if it had an editor that didn't make me want to throw the terminal through the wall"

when I used it . . . I'd rather use vi than goose quills, but it's a close race."

WYSIWYG vs. Straight ASCII

"Why would I need WYSIWYG to edit a program? If it does anything but edit straight ASCII, it's not for programmers."—Roy LeBan

One of the most recent developments in text editors has been the "what you see is what you get" philosophy, which maintains that what's on the screen should be as close as possible to the exact appearance of the text when it's output onto paper. With newer high-resolution graphics screens, this approach has become increasingly feasible. WYSIWYG does have its merits, especially when you're aiming for high-quality hard copy. There is, however, a price to pay. The processing power and memory required for WYSIWYG editing frequently result in editors that are too slow and cumbersome for a large number of users.

The WYSIWYG approach has been married to what Steve Jasik, author of MacNosey, refers to as the "point and grunt interface"—that is, the mouse and pointer method. This has perhaps been the most revolutionary change in text editing on personal computers, but it's also been one of the most controversial ones. Not only does the heavily graphical WYSIWYG style slow down the editor, but also the mouse itself is seen by some as an unnecessary encumbrance. These people object to the need to move their

Wish List: The Ultimate Editor

We asked some programmers to list the features they would like to see in the ultimate editor and got many good, if not definitive, responses. This list came from Chris Dunford. It's important to realize that these are (and must be) personal opinions. Value judgments are all important when evaluating editors, and your own experience may be vastly different. Feel free to make up a list of your own and send us a copy—we'd be interested to hear more from our readers on this topic.

Must Haves

- Line orientation.
- Reconfigurable keyboard.
- Macros that must be able to make decisions, not just remember keystroke sequences, and/or programming language. Must be able to assign macros to keys.
- Line/block-move/copy/delete (one-dimensional). Optional two-dimensional blocks—both character stream and arbitrary-rectangle blocks.
- Global search/replace, case-insensitive option, with wildcards. Search/replace should be restrictable to specified portions of the text, including column orientation (replace *foo* with *boo* over lines n-m from columns x-y).
- Shell to operating system.
- No menus, or at least the ability to run in command mode and bypass menus. Command mode should not just be obscure keystrokes.
- Multifile capability with interfile operations (for exam-

ple, move text from one file to another and perform operation x on all files) for at least six files.

- Tab stops user-definable, and the option to use tab compression when writing to disk.
- Optional auto-indent (OK if done via macro).
- Internal operations must be fast.
- Must have "go to line n" or "go to current line + n" and be able to mark locations in the file for *gos*tos.
- Some undo capability—for example, "restore last n lines that were altered."
- Read/write blocks from disk (for example, insert file x at cursor position; write marked block to disk file y).

Not Necessary for Everybody

- Multiwindow operations with configurable window sizes (should be capable of both top-to-bottom and side-by-side windows).
- Block left/right shifts.
- Text overlay operations.

That'd Be Nice, but I Can Live Without It

- Edit files larger than memory.
- Some optional word processing features: line center, paragraph reformatting, case conversions, ability to tag groups of lines.

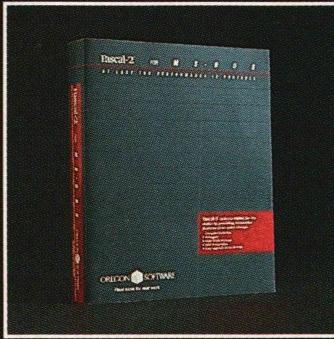


TALK OF THE TOWN

One language supports this community.
That language is Pascal-2, now on the PC and producing the fastest, most compact code available. For the professional programmer, imagine what you can do with this power:

- Cut execution time by 20% to 200%
- Transport MS-DOS programs to VAX, PDP-11, and 68000 machines with only minor adjustments
- Cut executable program size by up to 50%
- Use all of DOS-addressable memory through efficient large-memory model
- Speed error correction and save development turn-around time with sophisticated error checking and reporting
- Find and fix logical errors with the interactive source-level debugger
- Access DOS services

Pascal-2™ FOR MS-DOS



and network files ■ Call Microsoft FORTRAN, C, Pascal, and assembler
■ Upgrade from TURBO Pascal with compatible strings, equivalent procedures and access to TURBO graphics.

Plus!

- Intel CEL87 mathematical library for scientific computing
 - A special interface between Pascal-2 and the programmable BRIEF text editor (editor optional).
 - Certified ISO standard Level 1.
 - Dramatically improve your productivity and introduce your PC software to the VAX next door.**
- Call or write OREGON SOFTWARE, INC.
6915 SW Macadam Avenue,
Portland, OR 97219 (800) 367-2202
TWX: 910-464-4779 FAX: (503) 245-8449

OREGON  SOFTWARE

Real tools for real work

AT LAST THE PERFORMANCE IS PORTABLE

The following are trademarks: Oregon Software, Pascal-2, Oregon Software, Inc.; IBM, PC-AT, PC-DOS International Business Machines Corporation; Intel, Intel Corporation; MS, Microsoft Corp.; TURBO Pascal, Borland International, Inc.; BRIEF, UnderWare Corp.; PDP, VAX, Digital Equipment Corp.

Circle no. 357 on reader service card.

TEXT EDITORS

(continued from page 18)

hand frequently from the mouse to the keyboard and back.

And then there are those who prefer to have it both ways. Dennis Brothers says: "My workhorse editor these days is the Macintosh Programmer's Workshop (MPW) shell. It functions as both an editor and the command language interface for MPW, with all the advantages of both a Mac-like 'point and click' system and a programmable, command-oriented editor. You ought to see me fumbling for the mouse when I have to edit something on a PC!"

The straight (non-WYSIWYG) text editors seem to fall into two categories. On one hand, you have the WordStar-style editors, which use lots of special control characters to add power and speed but sacrifice simplicity (and usually use "weird" file formats that can't be read by any other editor without some sort of conversion). On the other hand, you have straight ASCII editors such as XY-Write, PC Write, and Apple's MDS editor, which always keep the text in its purest form and generally display every single character of the file on the screen, including control characters. This is very useful when uploading files to computer networks or writing something that will be sent to a typesetter. Such editors often have a command line that is separate from the text window or a menu bar that fills the same function.

Kibitz Mode

"The so-called syntax checking editors are good for about ten minutes, then you learn that only the people that spec'ed them actually code that way."—Don Watkins

In the last few years a new breed of editor has sprung up. These new editors are supposed to ease the job of

software designers, chiefly by performing real-time syntax checking of the source file being edited. They tend to be very specialized, requiring a different version for each particular dialect of a language. These language-oriented editors might be great for relatively inexperienced programmers, who can often save a great deal of debug time by doing a syntax check without even leaving the editor. But in our experience, expert programmers tend to prefer the simplicity of a straight ASCII editor.

Chris Dunford, software author and a sysop on IBM NET, says: "The current smart editors are so restricting that it's like wearing a straightjacket. I have a friend who calls them 'Nazi' editors. They tell you what you can and can't do, and that's the wrong attitude. A smart editor should be one that watches over my shoulder while I use it free form (that is, just like I would use a dumb editor) and figures out what I am doing and does helpful things—[HAL-like voice:] Dave, I don't see a declaration of that variable you just used. Would you like me to declare it for you?—But if I want to do something it thinks might be wrong, well, by God it's gonna let me do it. Let's face it. With the current crop of smart editors, you have to tell them exactly what you're about to do. How smart is that?"

Again, for programmers who prefer using one editor for all their work, whether it's source code or documentation, this kind of editor is not a viable option. There seems to be another disadvantage as well. Dennis Allison mentions a syntax editor called Program Synthesizer that was at one time used at Cornell University: "[It] would not let you write a bad program. When the students tried to write a program using Emacs, they couldn't write one that worked." Of course, Allison also notes that he likes a couple of the syntax-driven editors... if the syntax-check mode can be turned off.

Do What I Say

An issue that is especially pertinent to programmers is

On-Line Editing

With the increase of telecomputing traffic in recent years, another important editing issue has arisen. At first, there was little need for on-line text editing. Messages were usually short and concise, mainly because on-line time was expensive. Now, though, electronic mail has become more and more sophisticated, and with it has come the need for quality on-line editors. The problem, simply stated, is that there is no standard for any high-level on-line interface. The current default for almost all systems is something called TTY, a relic from the days when all terminals were hard-copy printers. Under the TTY interface, there is no way to go back to a previous line and change it. Once something is printed, it's immutable. So any editors have to be similar to the old TECO style. Given the TTY constraints, an admirable amount of progress has nonetheless been made in on-line editors. Several generally accepted standards have evolved for the command interface, and on-line editors today seem far more usable than the old TTY editors.

The final solution to the on-line problem really comes

in two forms: the first, and perhaps most obvious, is to edit the information off-line and then upload it to the on-line system in one pass. The other solution is more technically difficult but offers more power in the long run. It is to install a front-end program at the user end that implements a quality, full-screen editing interface and uses a defined protocol to send data back to the host. The main advantage of this over off-line editing is that the host's full database can be made available interactively during such an edit.

Editing for Uploading

Here the most obvious problem is compatibility—because you don't know much about the destination system, often not even what kind of computer it will be, you must usually send a completely clean ASCII file. Some front-end packages for the larger information services include their own off-line editors; others include a way to read in a text file and send it. In both cases the text is likely to be quite clean.

programmability. Some editors allow you to construct macros that substitute a whole chain of commands for one simple command (or keystroke). That's the first step. Then there are editors that allow you to create miniprograms, complete with looping control structures and if...then statements. Beyond that you have editors that can be interfaced to custom-designed program modules, written in C, assembly language, or some other language. The ultimate in programmability is perhaps best expressed by Dennis Allison at last year's Hackers Conference: "Just give me the source code."

Programmers tend to prefer programmability somewhat more than typical users do, but even among programmers, there's a wide range of preferences about this feature, too. Some go so far as to write their own editors, whereas others prefer not to be presented with so many choices in editor configuration, preferring to learn a given set of commands and to get on with the job at hand.

The Ultimate Editor?

Many programmers, unsatisfied with the state of the art, have created their own answers. Jef Raskin of Information Appliance Inc. has invented a special interface called the SwyftCard (see *DDJ*, June 1986, for a review).

Bob Wallace has this to say about the ultimate editor: "There are some identifiably different approaches that people have to the world in general—some people are visually oriented; others respond to things more in terms of touch, spatial relationships, or sound. The thing about text editors and word processors is that there is room for all these user interfaces, and there is no reason why one editor can't be accessible from more than one approach."

Summing It Up

When asked to give his views on text editors, former *DDJ* columnist Dave Cortesi deftly sidestepped specifics and instead gave us the following parable.

"IBM has this big internal network, VNET, that everybody uses to swap messages and files. And there is, or was, a newsletter, *VNET News*, distributed to hundreds of users around VNET each month. About this time there was a great proliferation of editor programs around IBM; everybody had their pet editor and wouldn't look at anybody else's. The NIH (not invented here) factor was fierce.

"So I was young and stupid, and I wrote a letter to the editor of *VNET News*, saying essentially, 'It's all malarkey. We don't need any more editors, let's quit experimenting with these trivial variations and use (the editor I preferred at the time).'

"Nobody could agree on the best editor, but there was one thing that got universal agreement: I had the wrong attitude, and the one important thing was to keep experimenting. Boy, did I get rebutted! From which I learned, never discuss religion, sex, or editors with anybody—especially editors."

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 2.

IQCLISP

More Common Lisp features in less space
for less money than any other IBM-PC lisp.

- MSDOS portable
- Bignums, 8087 support
- Multidimensional arrays
- Full Common Lisp package system
- Full set of control primitives
- Keyword parameters, macros
- Save/restore full environments for speed
- STEP, TRACE, BREAK, DEBUG, ADVISE, APROPOS
- Roll-out frees space for invoking MSDOS commands

IQCLISP PACKAGE \$300.

sq Integral Quality

LISP

P.O. Box 31970
Seattle, Washington 98103
(206) 527-2918

IQLISP

Now with a compiler.

- Compiler comes with source
- Compiler cuts execution time 50-75%,
program size 60-80%
- Multidimensional arrays
- Floating point, bignums, 8087 support
- Macros
- Color graphics
- Multiple display windows
- Assembly language interface
- Available for IBM PC or TI-PRO

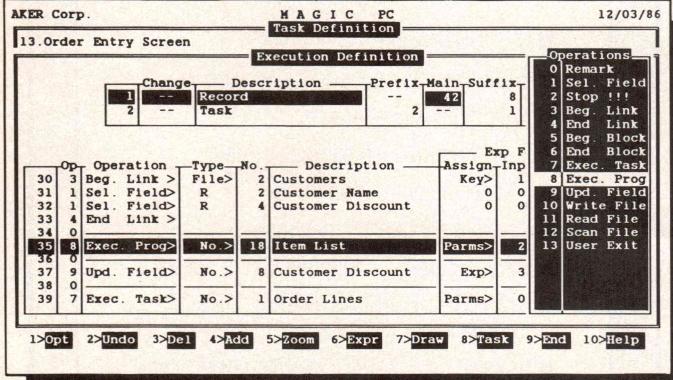
IQLISP PACKAGE \$270.

INCLUDES COMPILER

- VISA and Mastercard accepted
- Generous update policy
- Attractive educational license

Circle no. 327 on reader service card.

Announcing Magic PC – the first breakthrough for database applications developers in over 20 years: Now you can develop professional applications 1000% faster than your 4GL or DBMS, totally free from programming, commands and syntax!



A Magic PC program looks as simple as this. To design an application you quickly fill-in menu-driven decision tables **without having to write a single line of code**. For example, just by highlighting the Execute Program operation on this screen and also highlighting the Item List program in the Program Menu, you tell Magic PC to pop-up the Item List window shown in the adjacent screen, when the end-user hits the Zoom key.

Who needs another DBMS?

At last, Magic PC gives you the ultimate applications design tool, far ahead of 4GL's, DBMS and Application Generators.

Magic PC breaks through the language barrier with the revolutionary Un-Language concept:

NO PROGRAMMING, COMMANDS OR SYNTAX!

Free yourself from your programming language

Magic PC makes you, the professional, completely free from the drudgery of procedural programming. No more cryptic commands, syntax or unforgiving procedural structures, because Magic PC does all the programming automatically. There's your competitive edge. The rest is up to you...

The Professional Choice

Already an international success, Magic PC is a profit maker and career booster for DP Consultants, System Integrators, VARs, MIS professionals, System Analysts, Programmer Analysts and Software Engineers. If you design PC applications professionally, you can't afford not to Un-Language now.

IBM France: "IBM encourages this introduction and can not help but salute such evolution..."

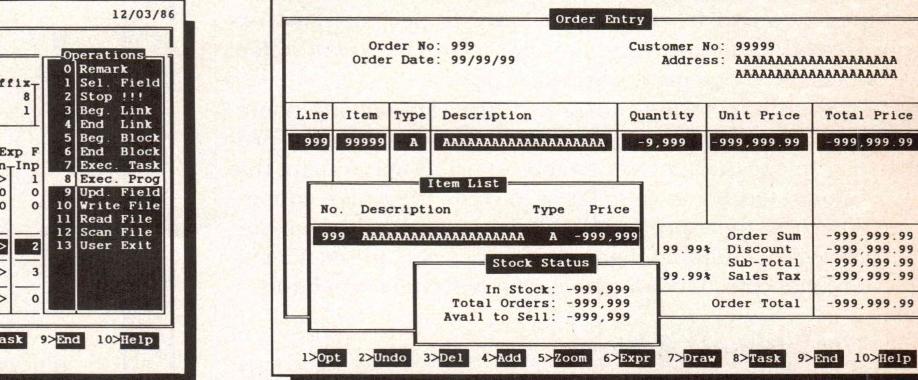
Israeli Air Force: "We were convinced that it was not possible to have a design tool powerful enough to implement real-life applications without a programming language. Magic PC changed our mind..."

Jeff Duntemann, PC Tech Journal: "It's probably the best integrated database applications and screen generator that I have ever seen... very smooth system, and smoothness comes at a premium these days..."

The Magic PC Secret

You're so much more productive with Magic PC because there is **absolutely no programming** to slow you down. You design a Magic PC application by simply filling-in the **Data Dictionary Tables** (Files, Fields, Keys) and the **Task Description Tables** (Operations and Expressions).

Only 13 design **Operations** harness the power of Magic PC. Operations are specific enough to eliminate the need for tiresome syntax, yet elastic enough to produce robust custom applications. Use the Operations to describe **what** you want and Magic PC makes it happen. It's that simple.



Magic PC gives your end-user the power to harness and retrieve data instantly, **without any commands or syntax** because at runtime you already have built-in options to Add, Delete, Modify, Query and get on-the-spot ad-hoc information simply by highlighting selections from menus. Data validation, security and error-checking are done automatically for you by Magic PC without programming.

Make Task nesting power available with a single **Execute Task Operation**. This powerful instruction triggers Magic PC to execute and display additional tasks or even external applications through **Window ZOOMS**. The 3-dimensional effect of Window Zooming lets you probe deep into your application through nested windows and manipulate the data underneath.

You describe a Magic PC Task or Program (composite Tasks) by filling your system analysis flow into the Task Description Tables. Choose the participating Data View, and Magic PC executes your desired Operations. You interface with the Tables by highlighting your selections from pop-up menu-driven windows. There's nothing to edit except your headings.

You're not confined to any particular design sequence as you are with most procedural languages. You can enter and change any Table spontaneously, on the fly, as ideas come to mind and Magic PC automatically maintains the application integrity.

A **Magic Inference Engine** automatically orchestrates your Task Description Tables into a single file of internal **Knowledge Base Rules** for optimum, bug-free performance. Knowledge Base Rules are executed by the **Magic Run** engine for stand-alone runtime operation, or by the **Magic LAN** engine for unrestricted Novell network sharing. You're free to design the Knowledge Base without worrying about the internal structure.

Discover fast,
language-free
programming
at no risk
for only

19.95



See for yourself how fast you can program language-free applications with our low-cost limited offer.

You'll get the full Magic PC software unprotected and limited to 100 records and 450 page documentation complete with a **free Order Entry sample application**. You'll also get our **free telephone support** for 90 days!

And your \$19.95 will be credited towards the full \$695 Magic PC purchase price. Even if you don't buy Magic PC right away, keep your \$19.95 Magic PC Trial as your application prototyping tool at this bargain price.

Our No-Risk Guarantee!

You have our no-risk 30-day money-back guarantee: if you're not completely satisfied for any reason, even Magic PC Trial for \$19.95, send it back for a refund.

Order now while supply lasts

Call this toll free number now with your Visa, MasterCard or American Express for immediate delivery, or send the Order Coupon below today to Aker.

**1-800-345-MAGIC
in CA call 714-250-1718**



Yes, please rush me:

- Magic PC Trial \$ 19.95
- Magic PC \$695.00
- Add shipping \$ 5.00
- In CA 6% tax \$ _____

Prices valid in US only.

Total \$ _____

Ship to: _____

Address: _____

City/ST/Zip: _____

Phone: _____

AKER

Aker Corp. 18007 Skypark Circle B2, Irvine, CA 92714
(714) 250-1718, Elec. Mail Dialcom 41:AKR 001 Telex 4931184
AKER OEM and VAR inquiries are welcome.

Min. requirements PC DOS 2.0, IBM PC or 100% compatible with 512K and hard disk.

© 1986 Aker Corp. Printed 1/87 **Trademarks:** Magic PC, Un-Language, Window Zoom, Magic Run, Magic LAN and Magic PC Trial are trademarks of Aker Corp., IBM and PC-DOS are trademarks of IBM Corp., Novell is a trademark of Novell Inc.



**"Microport is
the fastest path
between
real UNIX®
System V
and you."**

Dimitri Rotow runs a company on the go. He has little time for hardware or software that doesn't do the ultimate job for his company's customers.

Two months ago he assessed three readily-available UNIX (and UNIX-like) operating systems. After evaluating pricing, packaging, documentation, quality, service, manufacturer's support, compatibility, third-party support, features, conformity and performance, he and his VP of engineering came to one conclusion: "To praise Microport is to praise the AT&T/Intel joint venture, because Microport's version of UNIX System V is virtually identical. However, more value is received since more middlemen are removed—that's much closer to buying direct and selling direct."

Bell Technologies' customers appreciate real value plus real System V. If you're an OEM, a reseller, an end-user or simply curious about our product—call us today and see what all the endorsements are about.

Microport Systems, Inc.
10 Victor Square
Scotts Valley, CA 95066

408/438-UNIX (438-8649, local)
800/822-UNIX (Inside California)
800/722-UNIX (Outside California)
FAX: 408/438-2511
Telex: 249554



Team Use for The AT.

**Dimitri Rotow, President
BELL TECHNOLOGIES**
Circle no. 154 on reader service card.

UNIX is a registered trademark of AT&T

*See us at UNIFORUM;
Washington, D.C.;
January 20-23; Booth 1021*

6502 Hacks

by Mark S. Ackerman

Computers and controllers using the 6502 CPU often demand efficient use of both processing time and memory space. With an address space of only 64K, data and code space efficiency can be critical. Moreover, though advancements in these 6502 systems have allowed the use of high-level languages, there will always be a need for fast subroutines and tight programs.

This article is a result of writing code for an Atari VCS 2600 game unit that had only 128 bytes of RAM and 8K of ROM. Because of the limited graphics hardware, all processing was in real time with cycles counted. Several of the hacks and tips presented here are also transferable to other microprocessors. Many of the tips are appropriate to any true real-time system—that is, any system where real-time is measured in very small fractions of a second. My 6502 experience has been helpful in finding ways to crunch memory requirements and timing on the Intel 8086 and 80286.

In short, this article is a collection of my favorite hacks for the 6502. A fair amount of 6502 code is included, so the next section gives a brief introduction to the 6502. If you've programmed the 6502 extensively, you've probably been forced to memorize the instruction set in hex, so perhaps you should skip ahead.

The 6502

The 6502 has only five registers. The instruction pointer (*IP*) points to the

Mark S. Ackerman, 24 Chatham St., Cambridge, MA 02139. Mark is a computer researcher at the Massachusetts Institute of Technology, specializing in graphics environments.

***There will
always be a need
for fast subroutines
and tight programs.***

next instruction, as on most computers, and is not directly setable by the user. The accumulator (*A*) is the main register, and it is the only register with full arithmetic-logic unit (ALU) functionality. The two index registers, *X* and *Y*, are used for indexed addressing. The stack pointer (*S*) points to the top of the stack.

The 6502 also has a nonregular instruction set. For example, the *S* register can be set only from the *X* register using the *TSX* (transfer *S* register to *X* register) or *TXS* (yep—transfer *X* to *S*). Moreover, if you want to transfer a value between the *X* and *Y* registers, you must transfer through the accumulator or through memory. For example, a typical sequence might be:

```
TXA    ;tfr X to accumulator
TAY    ;tfr accum to Y
```

The 6502 also has several flags. The important ones are

- the negative flag (or minus flag), which is set when loading or doing ALU functions
- the zero flag, which is set in similar situations
- the carry flag, which is set in arithmetic operations
- the overflow flag, which is also set in arithmetic operations

There are also flags for enabling interrupts and for decimal mode.

Basic Hacks

One of the simplest ways to save code space is at initialization time. The following code, for example, might be used to initialize a few variables:

```
LDA #0      ;load accumulator
              with 0
STA FIRST   ;store accum in FIRST
LDA #2
STA SECOND
LDA #1
STA THIRD
LDA #0
STA FOURTH
LDA #5
STA FIFTH
```

This requires 20 bytes (2 per instruction) and a minimum of 25 cycles (2 per load immediate and 3 per store). The following would be cheaper:

```
LDX #0      ;X register=0
STX FIRST   ;store X in FIRST
STX FOURTH
INX         ;inc X by 1 (X now=1)
STX THIRD
INX         ;X now=2
STX SECOND
LDX #5      ;X now=5
STX FIFTH
```

This takes two cycles and 4 bytes less; you save 1 byte each per *INX* instruction. An even cheaper result can be obtained by:

```
LDA #5      ;accumulator=5
STA FIFTH   ;store accum in FIFTH
LSR A       ;shift right --
              ;(acc now=2)
STA SECOND
LSR A       ;acc now=1
STA THIRD
LSR A       ;acc now=0
STA FIRST
STA FOURTH
```

This last example uses the same number of cycles, 23, but costs only 15 bytes—a reduction of 25 percent over the first example. You might object that this is not “clean code.” Without adequate documentation, it may be less than clear, but it does save bytes and cycles.

This example also demonstrates a reduction principle: general reduction algorithms, such as using the index registers to increment instead of loading the accumulator with immediate values, can produce significant savings. The best savings, however, require a sharp eye for special situations.

Incidentally, if you use a loop during initialization, remember that the counter register contains -1 or 0 at the end of the loop. You can use this by-product for further savings:

```
LDA #\$C0
LDX #7
LP STA TABLE2,X ;put acc at TABLE2
; plus offset in X
DEX ;decrement X
BPL LP ;loop while X is
; positive (i.e., > 0)
INX ;at end of loop,
; increment X
STX ZERO ;store X (=0) in ZERO
```

Zero-Page Savings

The 6502's first 256 bytes of memory, zero page, have a unique property. Reads from and writes to zero page, including indexed I/O using only the X register, save a cycle and a byte. Frequently used variables, or memory registers, should be kept in this portion of RAM.

It is critical when addressing this memory to use the X register. Using the Y register for indexed addressing such as this:

```
LDA ZEROPAGE,Y
```

is actually an absolute addressed instruction—that is, the 6502 ignores whether the ZEROPAGE location is in zero page or not. The Y-indexed instruction uses an additional cycle for the fetch and a byte for the page address.

Using Left-Over Registers

Use all the registers. The index registers should be used for intermediate results. The following nonsense ex-

ample assumes that the stack is at \$FF:

```
LDX LOOP_COUNT
LOOP TXS ;store loop ct,
; freeing X
LDA WHATEVER
LSR A
TAX ;store acc/2
LSR A
AND #07 ;get low 3 bits
TAY ;get offset of TABLE
TXA ;restore acc
LDX TABLE,Y ;new index for X
ORA \$80 ;OR \$80 to low 3 bits
STA STORE,X ;put at STORE plus
; offset from TABLE + Y
TSX ;restore loop counter
DEX ;dec loop counter
BPL LOOP ;loop if counter >= 0
TXS ;restore stack ptr
```

Note that the TXS instruction can be used only if interrupts have been disabled. A temporary zero-page variable could have been used to replace the TXS and TSX at a cost of two cycles per loop execution and 1 byte.

Stack-Related Savings

It is often cheaper to place values onto the stack than to store them to temporary variables. A push (PHA) and pull (PLA) take seven cycles and 2 bytes. A store to zero-page memory with a following load takes six cycles and 4 bytes; a store to other memory takes eight cycles and 6 bytes.

When doing a substantial amount of I/O to temporary variables, it may make sense to actually reposition the stack pointer. This works only with page 1 variables.

Flags and the Bit Instruction

Careful use of bit flags can also save bytes and cycles substantially. The BIT instruction does a nondestructive test of a byte in memory. Bit 7 (the high-order bit) of the byte is placed in the negative flag, and bit 6 is placed in the overflow flag. No registers are affected.

For this reason, if RAM is limited, the two high-order bits of a byte are extremely valuable for Booleans. In fact, bit7 is valuable because it also sets the negative flag upon loading:

```
LDA WORD ;load acc with WORD
BPL NOT_SET ;go if + (bit7=0)
```

IS_SET AND #\$0F
TAX

In addition, the carry flag, which is set or cleared in shift operations, can be used to store a flag value temporarily. In this case, bits 7 and 0 are the most valuable.

If RAM is not limited, then a single Boolean in bit0 allows the use of the zero flag upon loading. The BIT instruction still cannot be used profitably unless the Boolean is in bit 7 or 6, however.

More Advanced Hacking

There are two ways to depend on preexisting conditions. The first is to assume that the carry bit is either set or not set as needed. In the 6502, the carry bit signifies just that: a bit is being set to indicate the carry. To add two 16-bit numbers, then:

```
CLC
LDA FIRST_VAR_FIRST_8
ADC SECOND_VAR_FIRST_8
LDA FIRST_VAR_SECOND_8
ADC SECOND_VAR_SECOND_8
```

If you know the condition of the carry, such as in the following sequence of instructions:

```
LDA FIRST
CMP #$18 ;comp acc to $18
BCS BRANCH1 ;go if acc >= $18
LDA VAR1
ADC VAR2
```

then a CLC can be omitted. Why? Because the branch was on a carry set condition, the only way into the addition would be if the carry were clear. In a similar manner, you can assume that there is no carry from a previous addition. For example, if VAR3 never exceeded 16 and VAR4 never exceeded 5, then the carry will never set. So the sequence:

```
CLC
LDA VAR3
ADC VAR4
STA TEMP1
LDA NEW1
ADC NEW2
```

can be used. The CLC for the second addition can be omitted because the carry will not be set. This, however, reduces the robustness of the code.

6502 HACKS

(continued from page 25)

Removing clear carries or set carries—used for subtractions—can save many bytes. You may need to use some ingenuity. If the carry is set, for example:

```
LDA TEMP
SEC #$FF ;subtract -1
```

you may want to add 1 to *TEMP* by subtracting -1.

Cheaper Branching

The second way to use preexisting conditions is with branching. If you know that a flag will be in a certain condition, the appropriate branch instruction can be used for an unconditional jump. This will save a byte but no cycles in the 6502; the unconditional *JMP* instruction takes 3 bytes whereas a conditional branch takes 2 bytes. Both take three cycles when the branch is made. (A conditional branch in which no branch is executed takes only two cycles.) For example, if the carry is set (perhaps from a *BIT* instruction as below or from a subtraction), then the code:

```
BCC JUMP_LOC
NEXT_LOC
```

forces an unconditional branch for the savings of a byte over a *JMP*.

Interestingly, the instruction sequence for a Boolean in *bit0*:

```
LDA YOUR_FLAG
AND #1 ;get bit0
BNE TRUE_SETTING ;branch on 1
FALSE_SETTING
```

can be replaced by:

```
LDA YOUR_FLAG
LSR A ;shift bit0 into carry
BCS TRUE_SETTING ;go if set
FALSE_SETTING
```

at a savings of a byte. This is especially useful for testing several bit flags in a single byte. It also nicely sets the carry bit for unconditional branching for both branches of an *if...else* structure.

```
LDA YOUR_FLAG
LSR A
```

```
BCS TRUE_SETTING
FALSE_SETTING ;carry is clear
some code
BCC END_IF
TRUE_SETTING
some code
END_IF
```

Table-Driven Code

You can make very large savings if you can replace code with preset data tables. Instead of attempting to compute divide-by-17s or sines, for example, it may be possible to have a table of the results for all expected values. For example, instead of computing *MOD7*, if the variable will never exceed 32, it will be far, far cheaper to have a table:

```
MOD7 DS 0,1,2,3,4,5,6
      DS 0,1,2,3 . . . etc.
```

Because many of these tables can be compressed or merged with other tables (as discussed later), the cost in bytes is reasonable. This method is certainly faster.

In a similar manner, decision tables, game-play paths, or timing decisions can often be decided prior to compilation rather than during execution. In games, it is often best to store the delta xs and delta ys instead of trying to compute sine wave patterns on the fly.

In many cases, you can use tables to speed up operations that are repeated often. If, for example, it is necessary to increment only the bottom nybble of a word, a normal addition cannot be used because the carry will ruin the top nybble. You could write:

```
LDA WORD
AND #$F0 ;get high nybble
STA TEMP ;store temporarily
LDA WORD
AND #$0F ;get the low nybble
CLC ;assume worst: clr carry
ADC #1 ;add 1
AND #$0F ;watch for wrap
ORA TEMP ;OR in high nybble
STA WORD ;store back out
```

This costs 24 cycles and 19 bytes. If you had a table:

```
NEXTINC DS 1,1,1,1,1,1,1,1
      DS 1,1,1,1,1,1,-15
```

the cost could be reduced to 19 cycles

and 13 bytes:

```
LDA WORD
AND #$0F ;get current low nybble
TAY ;index into NEXTINC
LDA WORD
CLC ;might not be needed
ADC NEXTINC,Y ;add, indexed
      ;by current value
STA WORD
```

If this calculation were done in many locations in the program, the table would quickly become much cheaper than the calculation. Incidentally, the add instruction could be replaced with a subtract (or even a logical OR) instruction. Your choice of which instruction to use might depend on what table you had lying around!

Unrolling Loops

One large trade-off between time and space is in unrolling loops. The loop:

```
LDA #1 ;outside the loop
LDX #4
LOOP STA LOC,X
      DEX
      BPL LOOP
```

can be changed to:

```
LDA #1
STA LOC
STA LOC+1
STA LOC+2
STA LOC+3
STA LOC+4
```

This is more costly in terms of bytes (12 bytes vs. 9 bytes), but it is far faster (17 cycles vs. 48 cycles). (The loop overhead takes $4*(2+3) + 1*(2+2)$.) It is often surprising how much time can be saved by unrolling simple loops.

It is also possible to combine loops, even of different sizes, saving the costly loop overhead:

```
LDA PICKUP+7
      ;move PICKUP's contents to TABLE
STA TABLE+7
LDA PICKUP+6
STA TABLE+6
LDX #5
LDY #$80
LOOP2 LDA PICKUP,X ;continue the
      ;move with loop
      STA TABLE,X
```

Now You Know Why **BRIEF** is **BEST**TM

"If you need a general-purpose PC programming editor, look no further. Recommended."

- Jerry Pournelle, Byte, 12/86

The Program Editor with the **BEST** Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)...is quite simply the best code editor I have seen."

MACRO LANGUAGE

As Steve McMahon describes in Byte (3/85), "BRIEF is even more customizable than [another customizable editor] - not only may wholly new commands be created, but they may be assigned to any key, replacing even basic function keys like cursor control keys or the return key... String and integer [variable] types are available and may have either global or local scope... Arithmetic and logical primitives, type conversion, buffer and window control, search and translate, keyboard input,...are [also] available.

"Much of BRIEF was written in the BRIEF macro language, and the source of these macros is included with the editor. Using this source, it's possible to customize even sophisticated functions of the editor..."

No other editor has a more powerful macro language.

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size -(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

Program Editing YOUR Way

A typical program editor requires you to adjust your style of programming to its particular requirements - NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key - even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion - **BRIEF IS BEST!**

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

MONEY-BACK GUARANTEE

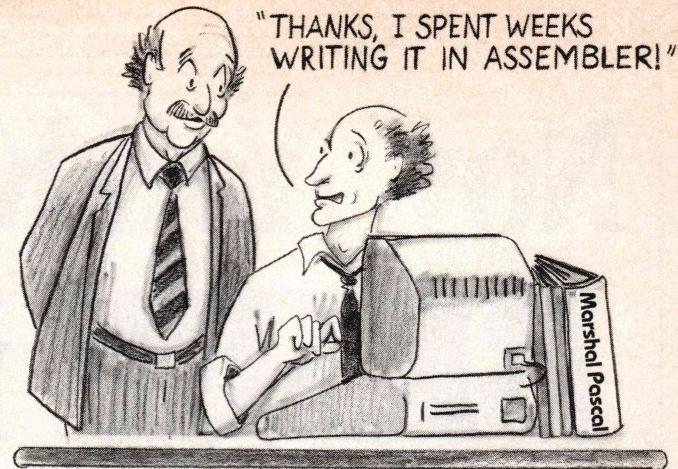
Try BRIEF (\$195) for 30 days - If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

**Solution
Systems™**

SOLUTION SYSTEMS, 335-D WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

BRIEF is a trademark of UnderWare

| Marshal Pascal | Ackerman | | Sieve | | I/O | |
|-----------------|----------|-----------|-------|-----------|------|-----------|
| | Sec. | Code Size | Sec. | Code Size | Min. | Code Size |
| IBM Pascal | 11.9 | 5.1K | 4.8 | 3.4K | 1.9 | 6.8K |
| Turbo Pascal | 12.4 | 34.7K | 11.7 | 27K | 2.6 | 28K |
| Microsoft C 4.0 | 22.7 | 11.6K | 14.2 | 11.5K | 2.2 | 12.5K |
| | 15.9 | 9.3K | 5.8 | 6.5K | 1.9 | 8.9K |



POWER CORRUPTS!

Actually, it only took him a couple of days with Marshal Pascal. Marshal Pascal produces code smaller and faster than any other High Order Language on the PC. Period. Even optimized C compilers don't come near our performance. And Marshal Pascal allows you to generate an assembly language listing of your code, to boot. So it's not surprising that some people would fudge the truth.

However, speed and compactness are not enough. Power and ease of use are important, too. With Marshal Pascal you may address as much memory as your operating system allows and a variety of memory models are supported. Among the useful extensions included are: separate compilation of modules (both Modula-2 and Pascal forms), structured constants and structured function values, variable-length string types, overlays, long integers and procedural parameters. Our relocatable linker allows you access to the Microsoft family of languages and assemblers. The data-flow analyzer in the compiler automatically crunches your code far better (and with less headaches) than one can achieve by manually utilizing register variables. We support the Intel 8087-80287 math processors *inline*. If you don't have the math chip, then '87/287 emulation is a provided option. We also provide a number of compile options, including an optimization by-pass for speedier compiles, I/O "fine-tuning," and constant folds. And a wealth of compile-time checks will help reduce your debug time enormously.

If that weren't enough, we also give you a **Turbo Translator**, allowing you to use the hundreds of application and utility programs written in Turbo Pascal® that are available in source from the public domain.

Space does not permit us to list all the features of Marshal Pascal, all of which are provided USER options, not additional COST options. Our compiler is not broken apart to make many products out of what should be only one good one.

Only, please folks, no matter how tempting it may be, don't fib to people once you start programming with Marshal Pascal!

the Price?
\$189.

Marshal Pascal™

Supports PC-DOS®, MS-DOS®, CP/M-86®, Concurrent Dos® and soon Xenix 286®!

Marshal Pascal is a TM of Marshal Language Systems, CP/M-86 and Concurrent Dos are ® of Digital Research, Inc., PC-DOS is a ® of IBM Corp., Turbo Pascal is a ® of Borland International, Inc., MS-DOS and Xenix 286 are ® of Microsoft Corp.

To order your copy of **Marshal Pascal** call: (800) 826-2222
In California: (415) 947-1000

Marshal Language Systems

1136 Saranap Ave., Ste. P, POB 2010, Walnut Creek, CA 94595

Circle no. 317 on reader service card.

Technical Product Information.

Dr. Dobb's Journal of Software Tools

Name _____
Title _____
Company _____ Phone _____
Address _____
City/State/Zip _____

February 1987 #124 Expiration Date: May 31, 1987

Please circle one letter in each category:

- I. My work is performed:
 - A. for in-house use only.
 - B. for other companies.
 - C. for end users/retailers.
 - D. in none of the above areas.
- II. My primary job function:
 - A. Software Project Mgmt/Sprv
 - B. Hardware Project Mgmt/Sprv
 - C. Computer Consultant
 - D. Corporate Consultant
 - E. Other
- III. My company department performs:
 - A. software development.
 - B. computer system integration.
 - C. computer manufacturing.
 - D. computer consulting.
 - E. computer research
 - F. none of the above.
- IV. This inquiry is for:
 - A. a purchase within 1 month.
 - B. a purchase within 1 to 6 months.
 - C. product information only.
- V. Corporate Purchase Authority:
 - A. Final Decision-maker
 - B. Approve/Recommend
 - C. No Influence
- VI. Personal Computer Users at my Jobsite:
 - A. 10,000 or more
 - B. 500 to 9,999
 - C. 100 to 499
 - D. 10 to 99
 - E. less than 10
- VII. On average, I advise others about computers:
 - A. more than once per day.
 - B. once per day.
 - C. once per week.
 - D. less than once per week.
- VIII. In my job function, I:
 - A. design software and/or write code.
 - B. design software.
 - C. write code.
 - D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding numbers at right for more info.

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 |
| 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 |
| 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 |
| 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 |
| 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 |
| 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 |
| 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 |
| 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 |
| 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 |
| 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 |
| 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 |
| 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 |
| 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 |
| 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 |
| 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 |
| 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 |
| 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 |
| 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 |
| 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 |
| 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 |
| 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 |
| 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 |
| 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 |
| 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 |
| 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 |
| 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 |
| 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 |

Circle 999 to start a 12 month subscription
at the price of \$29.97

TAKE THIS CARD WITH YOU
AS YOU READ THROUGH
THIS ISSUE OF DR. DOBB'S.



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

**Dr. Dobb's Journal of
Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157
Clinton, Iowa 52735-2157



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157

Clinton, Iowa 52735-2157

**TAKE THIS CARD WITH YOU
AS YOU READ THROUGH
THIS ISSUE OF DR. DOBB'S.**

| | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 1 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 1 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 1 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 1 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 1 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 1 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 1 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
| 1 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 |
| 1 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |
| 1 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 |
| 1 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 |
| 1 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 |
| 1 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 |
| 1 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 |
| 1 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 |
| 1 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 |
| 1 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 |
| 1 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 |
| 1 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 |
| 1 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 |
| 1 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 |
| 1 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 |
| 1 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 |
| 1 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 |
| 1 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 |
| 1 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 |
| 1 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 |
| 1 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 |
| 1 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 |
| 1 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 |
| 1 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 |
| 1 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 |
| 1 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 |
| 1 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 |
| 1 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 |
| 1 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 |
| 1 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 |

rcle 999 to start a 12 month subscription
the price of \$29.97

Dr. Dobb's Journal of Software Tools

Name _____

Title _____

Company _____ Phone _____

Address _____

City/State/Zip _____

February 1987 #124 Expiration Date: May 31, 1987

Please circle one letter in each category:

I. My work is performed:

- A. for in-house use only.
- B. for other companies.
- C. for end users/retailers.
- D. in none of the above areas.

V. Corporate Purchase Authority:

- A. Final Decision-maker
- B. Approve/Recommend
- C. No Influence

VI. Personal Computer Users at my Jobsite:

- A. 10,000 or more
- B. 500 to 9,999
- C. 100 to 499
- D. 10 to 99
- E. less than 10

VII. On average, I advise others about

- computers:
- A. more than once per day.
 - B. once per day.
 - C. once per week.
 - D. less than once per week.

VIII. In my job function, I:

- A. design software and/or write code.
- B. design software.
- C. write code.
- D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding

Technical Information Product

6502 HACKS

(continued from page 26)

```
STY TAB2,X ;set TAB2 to $80
DEX
BPL LOOP2
```

Chaining Subroutines

One of the simplest ways to save bytes is to use subroutines for common code. This requires the time cost of the *JSR* (six cycles) and the *RTS* (six cycles), however.

One way to save in subroutines is to create multiple-entry subroutines. That is, if two subroutines share a common ending, do not put that common ending in another subroutine. Instead, create a single subroutine. In a "pure" multiple-entry subroutine, you fall through all sections of code until the return statement. You can also jump around the noncommon code:

```
FIRST_ENTRY
first section of code
JMP END_SUB ;also try BCC
; or the like
SECOND_ENTRY
second section of code
END_SUB
final processing
RTS
```

The calling routines can call either *FIRST_ENTRY* or *SECOND_ENTRY* with their different processing. Both subroutine sections will exit from the same *RTS*, however.

Finding the Extra Microprocessor Cycle

Sometimes, in coding for real-time processing, you may need to kill an extra cycle or two. The 6502 has a two-cycle *NOP* (no operation) instruction. What about a single cycle? The 6502 has no single cycle *NOP*.

Of course, sometimes a single cycle isn't needed—only an odd number of cycles. You can get seven cycles by a combination of push and pull stack operations; five cycles can be bought by doing a *NOP* and a load from zero-page memory.

Single cycles can be gained only through other operations. One such operation is the absolute addressed load using an index register. Normally this instruction (*LDA ADDRESS,X* or *LDA ADDRESS,Y*) takes four cycles. If there

is a page boundary crossing (say that *ADDRESS* is at *C0F0* and the *X* register is 18), however, then the instruction takes five cycles. To gain the extra cycle, the table can be placed to force a page boundary crossing.

Occasionally you can use hardware memory mapping to the same effect. In the Atari VCS, for example, page 1 memory and page 0 memory were mapped together. Therefore, if *ZEROADD* were at *0065*, it could also be found at *0165*. A zero-page fetch costs three cycles, but the same fetch from page 1 memory costs four cycles.

It is also possible to branch to the next instruction depending on a flag. This kills either two cycles (for a non-executed branch) or three cycles (for a branch taken). This is occasionally useful for synchronizing *if...then...else* code.

Savings by Stepping Back

Perhaps the greatest savings can be had by proper planning. Putting two flags in bits 7 and 6 or in bits 0 and 1 makes more sense than putting them in bits 3 and 5. Often it is necessary to

make simple changes in the midst of programming in order to crunch code. This type of planning comes with experience.

Stepping back from the actual programming always helps. For example, you may have converted one data structure to another through easily understandable transformations, perhaps with intermediate data structures. If you are used to high-level languages in which this is encouraged, your assembler code will reflect it. Unfortunately, this type of elegance often turns out to be costly. The type of elegance that will benefit you in crunching will be the elegance of simple algorithms—almost always single-pass algorithms—that do not require special cases. Special cases cost.

Another type of planning that is often helpful is determining when program actions will occur. It may not be necessary to have all program functionality present at the same time. In a game, where time is critical, the *x,y* positions, for example, do not need to be updated every 1/60 second be-

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of version 1.1 of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is Bell Labs' answer to Ada and Modula 2. C++ will more than pay for itself in saved development time on your next project.

C++

from **GUIDELINES** for the IBM PC: \$195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.

Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for \$195:

- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:

send check or money order to:

GUIDELINES SOFTWARE
P.O. Box 749
Orinda, CA 94563

To order with Visa or MC,
phone (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.

Call or write for a free C++ information package.

Circle no. 351 on reader service card.

cause the human eye does not demand that. In business software, where space is more critical, you can overlay code.

Is It Really Necessary?

In addition, there is always a time to ask yourself, is that feature really necessary?

In a time-sharing scheme I was working on, for example, it was necessary to determine three-way time sharing. The ideal would have been to have a scheduling sequence such

as:

1 2 3 1 2 3 1 2 3 1 2 3

and so on, with each of the three actions (placing a graphics sprite on the display) getting one-third of the time. This order was needed rarely, however, the usual case being either two-way time sharing or no time sharing. Divides-by-3 are extremely expensive on the 6502, so I came up with this alternative:

1 2 3 1 1 2 3 2 1 2 3 3 1 2 3 x

where x was a skipped slice. It turned

out that the result was not significantly distinguishable, which saved many bytes and much time. (This could be implemented by ANDing a timer with \$07 and using a lookup from a table.) This approach transformed a difficult computation into a divide-by-4 problem.

All of this is not to say that design can be separated from coding. The inspired moment of coding often finds the necessary time and bytes when all the planning weeks (or even months) ago failed. It takes patience and skill to notice that TABLE2 can be created by taking the TABLE1 entry, exclusive OR-ing \$7F, and adding 5. It takes the same patience and skill to rip out a section of code and rearrange it to get the same overt behavior.

Killer Hacks

When all else fails, there are always a few tricks left. The following hacks are not for novices, though. These methods squeeze the final cycles and bytes from your program. They make debugging your code nearly impossible, and you might as well forget about maintaining the code later.

Please don your safety goggles.

Chaining Branches

One ugly way to reduce the number of bytes of code is to chain branches. As mentioned earlier, the 6502 uses 2 bytes for a relative branch instruction and 3 bytes for an absolute jump instruction. Unfortunately, the relative byte instruction can address a space of only 127 bytes forward or backward. Therefore, the unconditional JMP instruction is often used, even for implementing if... then ... else structures. It is possible to convert these JMPs to conditional branches, saving bytes.

If a condition is known, such as the carry bit being set or the overflow flag being clear, it is possible to branch to another branch. By chaining branches, it is possible to have conditional branching of more than 127 bytes distance. This is recommended only when it is important to conserve space at the cost of execution time—two branches have to be executed—and maintainability.

Self-Modifying Code

Self-modifying code can be used

New from Lifeboat:

MULTI-TASKING C++ LINKER

ADVANTAGE C++

Brings the power of C++ to your PC.

- Opens the door to object-oriented programming
- Allows programs with greater resilience, fewer bugs
- Fully compatible with existing C programs
- All the benefits of C without its limitations

ADVANTAGE Link

Everything you've always wanted in a PC-DOS linker.

- The fastest, most powerful PC-DOS linker available
- The first linker to take full advantage of extended memory
- Accepts Microsoft and Phoenix command files
- Supports up to 53 commands—more than any other linker
- Compatible with Microsoft CodeView

ADVANTAGE LIBRARY SERIES

TimeSlicer

Multi-tasking library streamlines C programming.

- Perform concurrent tasks and real-time event processing
- Includes header files for both C and assembly language and example programs with source code
- Compatible with C++ and object-oriented programming
- Critical resource management assured

To order or to obtain complete specification sheets, call:

1-800-847-7078 In NY: 914-332-1875

55 South Broadway
Tarrytown, NY 10591

LIFEBOAT

The Full-Service Source for Programming Software.

Circle no. 118 on reader service card.

profitably in critically real-time routines when literally every cycle counts. Instead of loading a loop counter or some such from a zero-page variable, you can just change the *LDX* immediate instruction on the fly. This saves a cycle.

In addition, instead of performing a load or store indirect indexed, which uses 2 bytes of zero-page RAM:

```
LDA (INDIRECT),Y
```

you can modify the destination address on the fly and perform an indexed absolute load or store to save a cycle:

```
LDA ADDRESS,Y
```

This, of course, will have you barred permanently from any MIS employment for the rest of your life. Actually, I could not use this on the VCS because of the lack of RAM, so I am not familiar with its difficulties. I have been told, however, that with adequate documentation, this sort of treachery can be maintained.

Use of the NMI Interrupt

Another nasty trick to save bytes is the use of the break (perform interrupt) instruction. The *JSR* (call subroutine) instruction requires 3 bytes per call, whereas the *BRK* instruction requires only a single byte. This method, however, requires that you not be expecting nonmaskable interrupts.

To use this method, you must set the *NMI* vector to the address of the most frequently used subroutine. The *BRK* instruction can then be used to call the routine. *BRK*, however, not only places the return address onto the stack but also pushes the flag byte onto the stack. If you do not return information in the flags, you can return from the interrupt with an *RTI*. If you need the flags that were set in the subroutine, however, return with:

```
PLA ;pop caller's flags  
RTS ;return normally
```

Unfortunately, the *BRK* instruction takes seven cycles instead of the *JSR*'s six. If a *PLA* is needed, that will also add four cycles. After you set the *NMI* vector (at a cost of 2 bytes), however, each call will save 2 bytes.

Overlapping Code and Data

You can squeeze data table space in two ways. The first sounds more difficult to maintain but actually turns out to be easier in practice. This is to find the appropriate data table values in your code space. For example, you might need the following flag table:

TABLE \$80,00

If you will be testing only *bit7*, however, the following table would also do:

TABLE \$A9,00

This just happens to be a *LDA #0* (load

immediate of 0) instruction. Finding the appropriate code in your program:

TABLE LDA #0

also eliminates the 2 bytes from your data space. Although you might want to comment this table extensively to remind yourself of what you did, the only real problem you will have is to find the table again. Note that this technique generally works only with small tables.

The second method is much more effective in finding bytes. If you have four data tables:

FEATURES:

- Optional strong type checking
- Overloading of function names and operators
- Optional guaranteed initialization of data structures
- Data abstraction
- Dynamic typing (virtual functions)
- Optional user-defined implicit type conversion

• Works with your present C Compiler

- Functions as a Pre-processor Translator — handles regular C code with no changes
- Type-checking and other features are optional — you can turn them off
- Already thousands of users at commercial sites
- Complete documentation: *C++, A User's Guide* by Bjarne Stroustrup of AT&T (Addison-Wesley, 1986)

The only commercially-available C++ customized to operate on PC's, micros, minis, and mainframes with popular C compilers, including:

VAX C
LATTICE
MICROSOFT

GREEN HILLS
WIZARD
WHITESMITH'S

We Specialize In: Cross/Native Compilers: C, Pascal, FORTRAN, Ada, LISP — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Profilers — OA Tools — Design Tools — Comm. Tools — OS Kernels — Editors — VAX & PC Attached Processors and more
We Support: 680xx, 80x86, 320xx, 68xx, 80xx, Clipper, and dozens more

A DIVISION OF XEL



60 Aberdeen Ave., Cambridge, MA 02138 (617) 491-4180

Designer C++ is a joint trademark of XEL, Inc. and Glockenspiel, Ltd of Dublin. Ada is a trademark of the U.S. Government (AJPO)

Circle no. 254 on reader service card.

6502 HACKS

(continued from page 31)

TABLE1 DS 0,1,2,3,0,0,0
DS \$80,3,2,1,0

TABLE2 DS 3,2,1,0

TABLE3 DS 0,0,0

TABLE4 DS 1,0,6

they can be profitably reorganized as:

TABLE1 DS 0,1,2,3
TABLE3 DS 0,0,0
DS \$80
TABLE2 DS 3,2
TABLE4 DS 1,0,6

Confused? TABLE3 and TABLE2 are now completely contained in TABLE1. TABLE1 also extends throughout most (but not all) of TABLE4. Note that this has saved 9 bytes from the original 22 bytes.

Often the program can be altered slightly to increase this type of savings. Using the nibble-increment example from the section on table-driven code, the choice of instruction used might also depend on what tables are available for merging. It is also possible to find the table backward within another table: your indexing must then proceed backward also, decrementing instead of

incrementing.

Similarly huge savings are almost always possible. If you have a bug and one of the tables must be changed, however, it will be extremely difficult to separate the original data tables without adequate documentation. I have always used this technique last.

Code as Data

There are particular instances—especially when there is absolutely no room left—when code can substitute completely for data tables. This is especially effective for simulating random movements, such as for a self-play mode (called the attract mode in games). Code for the 6502 tends to be somewhat heavy toward having bit7 set, but otherwise it can create effective random tables. It is often necessary to try many code sections, however, for the desired effect in the software action.

The Endless Trade-Off

The crunching process revolves around the standard trade-off of time vs. space. Even a simple change, such as removing redundant code, requires this trade-off. Subroutines have extra overhead and slow processing down. In-line code, as with macros, can greatly speed up critical processing but at the cost of an enormous code space.

Many of the hacks described here require this trade-off. Most are tricks that sacrifice one for the other. The other hacks all have the added cost of reduced maintainability or increased programmer effort.

When is it worth using these hacks? If you're writing Pascal code on the Macintosh or C code on the PC, they cannot help much in removing 30K from your 200K program. But if you're in a situation in which you need a fast interrupt routine, these techniques can help on any machine. They can also be useful if you need to reduce the code space for a desk accessory or a similar routine or if you're just the sort of person who gets excited by realizing that a flag can be reused.

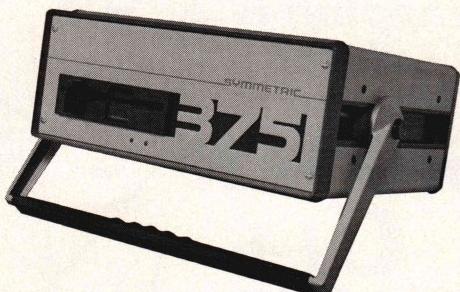
I'd like to thank the folks at General Computer for all their help.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 3.

SYMMETRIC
COMPUTER SYSTEMS

THE 375 - A SOFTWARE DEVELOPER'S DREAM



THE BEAUTY AND THE BSD

\$4,995*

OVER 1,000 PROGRAMS, 3,000 FILES !

Full 4.2BSD UNIX™

Each 375 comes standard with a complete configurable 4.2BSD UNIX system. We don't skimp on software. And we can even give you EMACS, INGRES, TEX and SPICE for those special applications.

Loaded With Languages

Turn on your 375 and start developing your own applications. It's that easy. C, FORTRAN, PASCAL, BASIC, APL, Assembler, LISP and PROLOG: they all come standard on every 375.

Loaded With Standard Features

- * 50MB Winchester Disc Drive
- * 1MB 5 1/4" Floppy
- * 2MB RAM (8MB optional)
- * Integral SCSI & ST506 Interfaces

- * 4 RS232 Ports (up to 36 optional!)
- * Parallel Printer Port
- * External Winchester and Floppy Ports
- * Series 32000, 10MHz, VM, FPU

Also Available: 10 Mb ETHERNET, up to 280MB disk memory, streaming tape, and more!

A Portable Computer for the Serious User

At last, a powerful, portable (22lbs) scientific computer for all your serious work. The 375 combines the advantages of a VAX™ with the size, versatility, and price of a micro. All the software tools are there. It's even small enough to leave on your desktop or take wherever you need it. **And it's available direct to you right now!**

CALL US TODAY (408) 279-0700

SYMMETRIC COMPUTER SYSTEMS - 1620 Oakland Rd. Suite D200 - San Jose, CA 95131

* Prepaid. Sales tax and Shipping Costs not included.
UNIX is a registered trademark of AT&T Bell Labs.

VAX is a trademark of Digital Equipment Series 32000 is a trademark of National Semiconductor Corp.

Circle no. 364 on reader service card.

Windows, Data Entry, Help Management, Menus, Text Editing, plus ...

SOURCE CODE

Vitamin C

It's good for your system!

The Vitamin C Difference

With **Vitamin C**, your applications come alive with windows that explode into view! Data entry windows and menus become a snap, and context sensitive pop-up help messages are nearly automatic.

With **VCScreen**, you'll save time by interactively painting windows and forms so what you see is what you get! Then, one button generates C source code ready to plug into your program and link with Vitamin C.

Easy enough for the beginner. Versatile enough for the professional. Vitamin C's **open-ended design** is full of "hooks" so you can intercept and "plug-in" special handlers to customize or add features to most routines.

Of course, Vitamin C **includes all source code FREE**, with no hidden charges. It always has. That means you'll have everything you need to adapt to special needs without spending hundreds of dollars more.

Windows

Create as many windows as you like with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, 4-way scrolling, scroll bars, sizes up to 32K, text file display & editing, cursor display, and more.

Unique built-in feature lets users move and resize windows during run-time via a definable key.

Access the current window by default or a specific window any time, even if it's hidden or invisible. Save and load windows on disk for more versatility!

Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields. Picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited data validation via standard and user definable routines. That means you aren't locked into one way of doing things.

Vitamin C even provides true right-to-left input of numeric fields with dynamic display of separators & currency symbols.

High Level Functions

Use our integrated help management, multi-level menus, and text file routines, or build your own handlers using Vitamin C's basic windowing and data entry routines.

Standard help handler provides context sensitive pop-up help messages any time the program awaits key strokes. The help text file is stored on disk and indexed for quick access. So easy to use that a single function initializes & services requests by opening a window, locating, formatting, displaying, and paging through the message.

Multi-level "Macintosh" & "Lotus" style menus make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens, quickly and easily.

Text editor windows can be opened for pop-up note pads, memo fields, or general purpose editing. Features include insert, delete, word wrap, and paragraph formatting.

30 Day Money Back Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C \$225.00

Includes ready to use libraries, tutorial, reference manual, demo, sample, and example programs, and quick reference card. For IBM PC and compatibles. Specify Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, Wizard, DeSmet, or Datalight C compiler AND compiler version number when ordering.

Vitamin C Source . . . FREE*
*Free with purchase of Vitamin C

VCScreen \$99.95
Requires Vitamin C and IBM PC/XT/AT or true compatible.

ALL ORDERS:

SHIPPING: \$3 ground, \$6 2-day air, \$20 overnight, \$30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on a U.S. Bank. Texas residents add 7 1/4% sales tax.

For Orders or More Information, Call ...

(214) 245-6090

creative
PROGRAMMING
Creative Programming Consultants, Inc.
Box 112097 Carrollton, Texas 75011

Hashing for High-Performance Searching

by Edwin T. Floyd

Programs that process symbolic information, such as compilers, interpreters, assemblers, spelling checkers, and text formatters, maintain an internal list of symbols or words—a symbol table. The speed of the symbol table's search and update operations often determines the performance of these programs. A hashing or scatter storage symbol table is easy to program and nearly always performs much better than a linear list or binary tree. In this article, I'll describe a technique called open hashing, discuss some of its performance factors, and then introduce a simple modification that can more than double the speed of the technique.

Open Hashing

Each symbol in a symbol table is represented by an identifier that is usually a string of alphanumeric characters, or a word. Each variable name in a Pascal program is an identifier, for example. An identifier is often associated with other data of interest to the application—for instance, an entry in the variable symbol table for an interpreter may be associated with a value for that variable, or an entry in the word table for a text analysis program may be associated with a reference count for that word. For now, don't worry about how to associate

A hashing symbol table nearly always performs better than a linear list or binary tree.

data with a symbol or even how to store strings of characters in memory. Suppose you can store the identifier string for a symbol somewhere in memory and retain its location as an index value, address, or pointer. Also suppose you can associate each identifier with a pointer to another identifier; thus you can form a list, or chain, of identifiers, each pointing to the next, until you reach the end of the list, symbolized by (*nil*). For instance, you might represent a chain of three identifiers by:

word1--->word2--->word3--->(*nil*)

An open hash table is a linear array of pointers called buckets, each pointing to the beginning of a list of symbol identifiers. For example, you might represent an open hash table with 101 buckets as shown in Table 1, page 35. Bucket number 0 points to the list of identifiers *left*, *help*, *replace*; bucket number 1 is empty; number 2 points to the list *down*, *word*, *print*, *align*; and so on. The bucket pointers are adjacent to one another in an ordinary array, but the identifiers may be scattered about in dynamic memory or located in a separate array and strung together by their pointers into a list.

How do you decide to which bucket a word belongs? You decide with a hash function. For example, you might add up the ASCII character val-

ues of the letters in the identifier string, divide by the size of the table, and use the remainder as the bucket number. (This is actually a widely used method and is one of the four I analyze later.) The remainder after integer division, or modulo (*MOD*), is always less than the divisor, so if you use the table size as the divisor, the remainder is always a valid bucket number. When you insert an identifier in the table, you first compute its hash number, which is the bucket number. If the bucket pointer already points to an identifier, this is called a collision, and you add the identifier to the list of symbols already associated with that bucket. If the bucket is empty, you simply point it to the new identifier. When you wish to search the symbol table for an identifier, you again compute its hash number, which determines the bucket containing the identifier if it is in the table. You then need to search only the list of symbols associated with that bucket. Using this method, on average you will search about 1 percent of the full list.

Pascal data structures for a hash pointer table and symbols are presented in Listing One, page 44. Listing Two, page 44, provides routines to initialize, insert, and locate symbols, and Listing Three, page 47, provides four different hash functions.

Performance

As part of a performance test, I analyzed a text file consisting of about 1,200 lines of Pascal code. The first pass through the file collected all unique identifiers in a 101-bucket hash table using the "sum of the characters" hash function described earlier. Out of 3,198 noncomment words,

Edwin T. Floyd, 4210 Pickering Dr., Columbus, GA 31907. Edwin heads the Data Processing Department of the Hughston Sports Medicine Foundation in Columbus. In addition to working on electromyographic signal analysis, he maintains a medical database with more than 130,000 patient histories that goes back 40 years.

243 were unique identifiers. The second pass searched the hash table again for each of the 3,198 words and simultaneously counted "probes," or comparisons of the search word with a symbol table identifier. The second pass counted 7,216 probes—an average of 2.26 probes to find any given word. This result is about what you would expect from 243 identifiers distributed to 101 buckets. In contrast, with the same data in a balanced binary tree or a sorted list, you would average about seven or eight probes to find each word, and in a simple linear list you would average more than a hundred.

Evidently, the number of probes necessary to find any given symbol appears to be approximately the number of identifiers stored in the table divided by the number of buckets. (Actually, for a uniform word distribution, you would expect it to be about half this number, but it's not because the word distribution is *not* uniform, as you shall see.) Therefore, one way to improve the performance of a hash table is to increase the number of buckets. With the same Pascal source file in a 203-bucket hash table, for example, you should more often than not find the search symbol on the first probe.

I also analyzed one quick-to-compute hash function—the sum of the first (times a constant) and last characters plus the identifier length.

Aho, Sethi, and Ullman¹ describe and analyze a hash function they call HashPJW (see Listing Three). In studying HashPJW, I was struck by its similarity to software routines for computing cyclic redundancy check (CRC) codes used to detect errors in disk storage and data communications. I suppose this makes intuitive sense—CRC code algorithms are designed to generate as many unpredictably different codes as possible for widely varying input data, with the hope that a packet or sector with an error will have a different CRC from the original and thus the error will be detected. A hashing algorithm based on CRC should do quite well on the average, and fast, table-driven routines exist in the public domain. One such routine is also presented in Listing Three.

A poor hash function can reduce performance by overusing some buckets and underusing others. A

hash function may be quite acceptable for one set of symbols and horrible for another. Aho, Sethi, and Ullman also describe a statistic (here called $U(h,t)$) that characterizes the uniformity of a hash function (h) in distributing a given set of symbols (t) to hash buckets. A routine to compute $U(h,t)$ is given in Listing Four, page 48; it computes the results for each hash function and text file discussed. The lower the $U(h,t)$ number, the more uniform is the distribution—a perfectly random distribution would have a $U(h,t)$ of 1.000, and a distribution more uniform than random would have a $U(h,t)$ less than 1.000.

In summary, the four hash algorithms I analyzed were:

• (sum of the characters + length)
MOD 101

- (first * 256 + last + length) MOD 101

- HashPJW MOD 101

- CRC-16 MOD 101

I analyzed three text files:

- a 1,200-line, 3,198-word Pascal program (comments and strings excluded) with 243 unique identifiers

- a 1,300-line, 9,025-word legal deposition outline with 1,068 unique words

- a 1,150-line, 5,516-word business report with 1,402 unique words

My results are summarized in Tables 2, 3, and 4, below. In general, the choice of hash function didn't appear to make a significant difference to the performance of the program. The difference in number of probes between the best performing and the worst performing hash function was never

```

0. --->left--->help--->replace--->(nil)
1. --->(nil)
2. --->down--->word--->print--->align--->(nil)
3. --->tab--->(nil)
4. --->(nil)
5. --->screen--->insert--->off--->format--->(nil)
6. --->find--->save--->(nil)

.
.
.
100. -->quick--->turn--->line--->(nil)

```

Table 1: Pointer identifier list

| Hash Method | No. of Probes | U(h,t) | Probes/Search |
|--------------|---------------|--------|---------------|
| Sum of chars | 7216 | 1.022 | 2.26 |
| F + L + len | 7611 | 1.127 | 2.38 |
| HashPJW | 6410 | 1.045 | 2.00 |
| CRC | 6925 | 0.981 | 2.16 |

Table 2: Pascal source code, average 2.40 identifiers per bucket

| Hash Method | No. of Probes | U(h,t) | Probes/Search |
|--------------|---------------|--------|---------------|
| Sum of chars | 81544 | 1.034 | 9.04 |
| F + L + len | 84702 | 1.178 | 9.38 |
| HashPJW | 82780 | 0.993 | 9.17 |
| CRC | 87147 | 1.014 | 9.66 |

Table 3: Legal deposition outline, average 10.57 words per bucket

| Hash Method | No. of Probes | U(h,t) | Probes/Search |
|--------------|---------------|--------|---------------|
| Sum of chars | 63036 | 1.004 | 11.43 |
| F + L + len | 72479 | 1.217 | 13.14 |
| HashPJW | 59807 | 1.001 | 10.84 |
| CRC | 59608 | 0.995 | 10.81 |

Table 4: Business report, average 13.88 words per bucket

THE DOCTOR MAKES HOUSECALLS!



Don't wait to hear the diagnosis from friends and co-workers . . . get it straight from the Doctor in your own home. Subscribe to **Dr. Dobb's Journal** and enjoy the convenience of having your personal copy delivered to your home or office each month.

And you'll save over \$5 off the cover price!

Every issue of **Dr. Dobb's Journal** will bring you indispensable programming tools like algorithms, coding tips, discussions of fundamental design issues, and actual program listings.

You'll find regular coverage of:

- Popular languages such as C, Assembly, Forth, Pascal, Ada, Modula-2, BASIC, FORTRAN, and Cobal.
 - 68000 and 80x86 architectures
 - MS-DOS, and Unix operating systems
 - Usable techniques and practical applications of AI and object-oriented programming research.
 - New product reviews, and lively discussion of professional issues in software design.
 - Compilers, cross assemblers and much more!
- Dr. Dobb's Journal of Software Tools** . . . the magazine that has lived up to its reputation as the foremost source of technical tools since 1976. One year (12 information-packed issues) is just **\$29.97** - to subscribe simply mail in the attached card. But do it today . . . you won't want to miss *any* of the exciting issues we have planned!

**Dr. Dobb's Journal of
Software Tools**
FOR THE PROFESSIONAL PROGRAMMER

Dr. Dobb's Journal of Software Tools the Rx for programmers

SUBSCRIBE AND SAVE

Subscribe to Dr. Dobb's Journal
and save over \$5—a 15% savings
off the cover price!

Please charge my: Visa MC American Express
 Payment enclosed Bill me later

Card # _____ Exp. date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____

ONLY \$29.97

Canada & Mexico add \$10 for surface mail. All other countries add \$27 for airmail. All foreign subscriptions must be pre-paid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

SUBSCRIBE AND SAVE

Subscribe to Dr. Dobb's Journal
and save over \$5—a 15% savings
off the cover price!

Please charge my: Visa MC American Express
 Payment enclosed Bill me later

Card # _____ Exp. date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____

ONLY \$29.97

Canada & Mexico add \$10 for surface mail. All other countries add \$27 for airmail. All foreign subscriptions must be pre-paid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

Comments & Suggestions

Dear Reader,

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take the time to write a letter. This card provides you with a quick and easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

—Ed.

February 1987, #124

Which articles or departments did you enjoy the most this month? Why?

Comments or suggestions: _____

Name: _____

Address: _____

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 790 REDWOOD CITY, CA

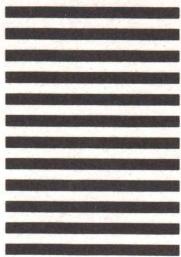
POSTAGE WILL BE PAID BY ADDRESSEE

**Dr. Dobb's Journal of
Software Tools**

FOR THE PROFESSIONAL PROGRAMMER

P.O. BOX 27809
SAN DIEGO, CA 92128

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 790 REDWOOD CITY, CA

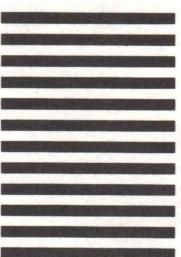
POSTAGE WILL BE PAID BY ADDRESSEE

**Dr. Dobb's Journal of
Software Tools**

FOR THE PROFESSIONAL PROGRAMMER

P.O. BOX 27809
SAN DIEGO, CA 92128

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



**Dr. Dobb's Journal of
Software Tools**

FOR THE PROFESSIONAL PROGRAMMER

501 GALVESTON DR.
REDWOOD CITY, CA 94063

PLACE
STAMP
HERE

**Dr. Dobb's Journal
of Software Tools . . .
the Rx for programmers**

HASHING FOR SEARCHING

(continued from page 35)

more than 23 percent. The second hash function did remarkably well considering the amount of information discarded by ignoring everything but the first and last characters and the length. This routine would be expected to do very poorly on identifiers that are all the same length and begin with the same character, such as those generated by some assembler macro processors. The CRC routine was unexpectedly worst on the deposition outline (long, but with not many unique words) but did very well on the others. The $U(h,t)$ function generally did pretty well in characterizing the search performance of the hash functions. I would be interested to hear the results of similar tests of these hash functions on other text files and of other hash functions.

A Self-Organizing List

Computer and human language text, particularly computer language text, do not contain a uniform distribution of words. If you were to count the occurrences of each unique word in a large sample of text and rank the words from most-often-used to least-often-used, you would find a wide disparity in word counts. You would also find a pattern: The product of the occurrence count and the rank order would be approximately constant. In 1949, George Zipf² described this word distribution pattern for several human languages and, oddly, populations of cities—it's now called Zipf's law. In addition, computer language text tends to be "clustery"—that is, references to the same identifier tend to be clustered together rather than uniformly distributed throughout the text. You can use this information to improve search algorithms.

Notice, after computing the bucket number with a hash function, searching an open hash table boils down to searching a linear list anchored in the chosen bucket. If you could organize the list so that the more likely an identifier is to be referenced next, the closer it is to the front of the list, the search time would be improved. If, every time you find a symbol during a search, you move it to the front of the list, the list will tend to organize itself in just

the optimum way for the clustery, unevenly distributed references that occur in human and computer language text.

The symbol search routine in Listing One requires seven additional lines of code to implement a self-organizing list with "move to front" (MTF). The modified search routine is

Computer language text tends to be 'clustery.'

presented in Listing Five, page 50, and the analyses of the four hash functions on the three text files are presented in Tables 5, 6, and 7, below. Clearly, the MTF optimization has a far greater effect on the number of probes than on any of the hash function variants. On Pascal source, the number of probes dropped at least 30 percent. On business report text, the drop was about 55 percent, and on the deposition outline, the drop was an incredible 66 percent or more.

In general, the more clustery and skewed the distribution, and the more items per hash bucket, the

more improvement MTF offers. I expect that Pascal source, arguably the most skewed and clustery of all text, would have shown more dramatic improvement had the average length of the word list at each hash bucket been 10 or 15, as in the other text samples, rather than 2 or 3. In the two human language text samples, the number of probes per search dropped to significantly less than half the average number of words per bucket. If words were distributed randomly in the text with uniform frequency, you would expect the number of probes per search to be about half the average number of words per bucket. Interestingly, with MTF the performance of CRC on the deposition outline text climbs into third place, well ahead of first + last + length and more in line with expectations. The $U(h,t)$ function remains unchanged because it depends only on the hash function and the text under analysis.

Conclusions

Hashing is a simple and effective organization scheme for symbol tables when rapid searching of the table is important but ordering (for example, alphabetically) is not. The choice of hashing function appears, on the basis of the tests I conducted, to have

| Hash Method | No. of Probes | $U(h,t)$ | Probes/Search |
|----------------------|---------------|----------|---------------|
| Sum of chars | 4326 | 1.022 | 1.35 |
| $F + L + \text{len}$ | 4552 | 1.127 | 1.42 |
| HashPJW | 4414 | 1.045 | 1.38 |
| CRC | 4550 | 0.981 | 1.42 |

Table 5: Pascal source code searched with MTF

| Hash Method | No. of Probes | $U(h,t)$ | Probes/Search |
|----------------------|---------------|----------|---------------|
| Sum of chars | 25659 | 1.034 | 2.84 |
| $F + L + \text{len}$ | 28043 | 1.178 | 3.11 |
| HashPJW | 25001 | 0.993 | 2.77 |
| CRC | 26079 | 1.014 | 2.89 |

Table 6: Legal deposition outline searched with MTF

| Hash Method | No. of Probes | $U(h,t)$ | Probes/Search |
|----------------------|---------------|----------|---------------|
| Sum of chars | 27882 | 1.004 | 5.05 |
| $F + L + \text{len}$ | 33314 | 1.217 | 6.04 |
| HashPJW | 27806 | 1.001 | 5.04 |
| CRC | 27802 | 0.995 | 5.04 |

Table 7: Business report searched with MTF

WRITE FASTER IN ANY LANGUAGE.

ASM
386, 286, 186,
86, 96, 51

intel

PL/M
386, 286, 186,
86, 96, 51

intel

C
386, 286, 186,
86, 96

intel

Fortran
186, 86

intel

If you develop software for any product based on an Intel microcontroller or microprocessor, including the 80386, the unique debug hooks in the Intel languages will help get the job done faster.

In fact, when used with Intel debuggers and emulators, Intel development languages can provide more debug data than any other high-level language.

Debug hooks let you

symbolically debug in the same high-level language you wrote in without having to deal with machine or hex code. Which means 80.386 reads as 80.386, not 50 62 D0 C5.

Because the location of both code and data are easily specified with our locator, it is easier for you to develop ROM-based firmware.

Since Intel languages

produce identical object code regardless of the host, you can write code at a PC running DOS, a VAX*/VMS terminal, or an Intel Development System.

Pascal 286, 186, 86



Software Debuggers

PMON 386, Pscope 86



Different members of the same

design team can therefore choose the most effective combination of languages and systems to get the job done faster.

Intel post-sales support can also help you get the job done faster. We invented the microprocessor. We know microprocessors and languages for Intel architectures better than anyone else.

When you buy an Intel language, you have access to our customer hotline. So if you ever have a question you can talk directly to a trained

Utilities

Relocator, Linker, Librarian



AEDIT

Programmers' Editor



applications specialist who understands our products. And can give you the right answers. Faster.

To order today, or get more information—including a free catalog of our development tools—call toll-free 1-800-87-INTEL.

The sooner you call, the faster you'll get the job done.



© 1986 Intel Corporation

*VAX is a registered trademark of Digital Equipment Corporation.

Circle no. 179 on reader service card.

little effect on the overall performance of the search, with the caveat that it is always possible to deliberately or accidentally contrive data that will defeat any hashing function. I would tend to favor HashPJW or CRC, which seem to be more difficult to fool. The incorporation of MTF self-organizing lists greatly improves the performance of the linear search phase at minimal coding cost.

Some Topics for Further Investigation

1. It's difficult to believe that the choice of hashing function has as little effect on performance as I observed. I would like to see the test results of the four hashing algorithms on a wider variety of textual material in human and computer languages. I can be reached at CompuServ ID [76067,747] or by U.S. Snail. Do you know a hashing function that consistently does better on a wide variety of text materials than the four presented here?

2. Hester and Hirschberg³ point out that MTF is not the only way to implement a self-organizing list. Some research (mostly assuming uniform distributions) indicates that exchanging the found item with the one preceding it on the list works better than MTF as the number of searches becomes arbitrarily large. How does this method perform compared to MTF with practical text files?

MTF and Exchange are opposite ends of the general "move ahead n" method (Exchange is move ahead 1, MTF is move ahead all the way). Is there some n, perhaps a function of the number of symbols on the list or the nearness of the found symbol to the front of the list, that produces better results than MTF?

Wild tangent: Huffman data compression depends on a distributed ordering of words or characters. Can a self-organizing list be used in a Huffman-like data compression routine that makes only one pass on the file?

3. If the identifier or search key is very long and the number of collisions is high, the time required to compare keys may become significant or even dominant. Morris⁴ suggests computing a hash code at least

three times longer than needed, using part of the code to determine the bucket number, and storing the remainder in the table instead of the identifier. The search routine then compares hash codes only, not the full identifier. This admits the possibility of a false match, but Morris believes the probability is acceptably low (which seems to me to be a blatant challenge to Murphy!). This has the attraction of eliminating the need to store a variable length character string. Do you have any experience with such a scheme, and would you be willing to share it?

4. What if you had a small list of identifiers that you needed to detect very rapidly in a text file—for example, the list of reserved words in Pascal? Is it possible to create a "perfect" hash function in which each identifier would hash to a unique bucket with no collisions and the number of buckets would equal the number of identifiers—or at least be a minimum? According to Sager⁵ it is, and the creation of such a function can even be partially automated. I would like to hear about experiences with Sager's algorithms implemented and used on a microcomputer. He reports creating minimal perfect hash functions for lists of up to 512 identifiers using an IBM 4341; what is the practical maximum for an IBM PC?

5. I've discussed hashing as a RAM-resident symbol table implementation technique. It has also been used as a disk database indexing technique. How is hashing on disk different from hashing a RAM symbol table? Is MTF optimization or any other kind of self-organizing list appropriate for disk hashing?

Listing Notes

All listings are in Turbo Pascal (Version 3, any operating system) for accessibility. The algorithms are almost trivial and should be readily convertible to assembly or any higher-level language. These routines use the following nonstandard Turbo Pascal features, which I feel serve to enhance the clarity of the code with minimal impact on its portability:

Declaration: *STRING [n]*—variable length string variables, equivalent to *ARRAY [0..n] of CHAR* where element 0 of the array contains the length of the

string. Assignment, concatenation, and comparisons are permitted on string variables.

LENGTH (x)—returns the length (0..255) of the string parameter; equivalent to *ORD (x[0])*.

COPY (x, i, j)—returns substring of string *x*, beginning with character *i* and continuing for *j* characters or until the end of string *x*.

SIZEOF (x)—pseudofunction returning the size in bytes of a variable or declaration. A constant value may be substituted.

GETMEM (ptrvar, len)—akin to *NEW*, returns a pointer to a chunk of dynamic memory of the specified length.

Infix Boolean bitwise operators: *AND*, *OR*, *XOR*—recognized as bitwise operators when used with integer operands and bit shift operators (*SHL*, shift left; and *SHR*, shift right).

Listing Two

The procedure *Symbol_Put* in Listing Two, which inserts the symbol name and data in the table, does not check for duplicate symbols. If you want to disallow duplicate symbols, use something such as:

```
IF NOT symbol_get (...) THEN
    symbol_put (...);
```

On exit, *this_symbol* points to the heap copy of the symbol just inserted, and *s_data* has been copied into *this_symbol^.sym_data*.

Symbol_Put attempts to conserve heap space by obtaining only enough memory to hold the identifier and its associated data, not the unused portion of the *sym_name* string. The technique used here depends on nonstandard features and characteristics of Turbo Pascal, Version 3, and may not be directly portable to other languages or future versions of Turbo Pascal. Nevertheless, I believe it to be the clearest exposition of the algorithm and defend it on those grounds.

Notes

1. A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers* (Reading, Mass.: Addison-Wesley, 1986), 434–436.

2. G. K. Zipf, *Human Behaviour and the Principle of Least Effort* (Reading, Mass.: Addison-Wesley, 1949).

3. J. H. Hester and D. S. Hirschberg, "Self-Organizing Linear Search," *ACM*

Computing Surveys, vol. 17 no. 3 (1985): 295–311. A survey of the current state of knowledge on self-organizing searches, particularly with Zipf distributions.

4. R. Morris, "Scatter Storage Techniques," *Communications of the ACM*, vol. 11 no. 1 (1968): 38–44 or vol. 26 no. 1 (1983): 39–42. The earliest description I found of open hashing, here called "scatter index tables." It also describes "virtual scatter tables."

5. T. J. Sager, "A Polynomial Time Generator for Minimal Perfect Hash Functions," *Communications of the ACM*, vol. 28 no. 5 (1985): 523–532. Graph/network-theory related algorithms for finding perfect hash functions.

Bibliography

Bentley and McGeoch. "Amortized Analysis of Self-Organizing Sequential Search Heuristics." *Communications of the ACM*, vol. 28 no. 4 (1985): 404–411. This paper indicates a preference for MTF for linked list implementations.

Knuth, D. E. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Reading, Mass.: Addison-Wesley, 1983. On page 514 Knuth describes open hashing as "collision resolution by chaining" and suggests using a self-organizing list but offers no implementation. Page 397 discusses Zipf's law and other probability distributions; pages 398–401 describe self-organizing files.

Peterson, W. W. "Addressing for random access storage." *IBM J. Research and Development*, vol. 1 no. 2 (1957): 130–146. This is generally credited as the first scientific paper on hashing, though Knuth reports finding references as early as 1953.

Availability

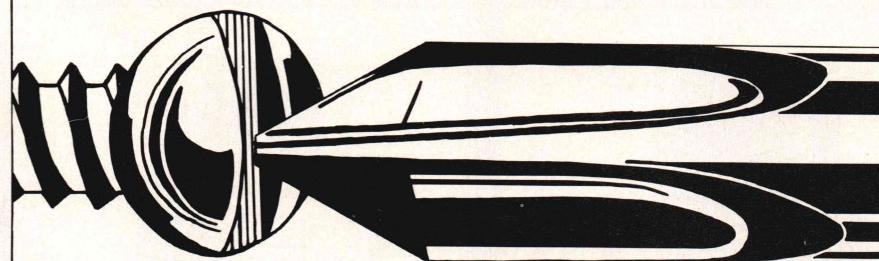
All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and disk format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 44.)

Vote for your favorite feature/article.
Circle Reader Service No. 4.

ISN'T IT A PITY...



Everything Isn't As Accommodating As

c-tree™ / r-tree™

FILE HANDLER

REPORT GENERATOR

Performance and Portability

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1, for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

r-tree: Multi-File Report Generator

r-tree builds on the power of **c-tree** to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to

create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

Unlimited Virtual Fields; Automatic File Traversal

r-tree report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, **r-tree** automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. **r-tree** even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royalty payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

How To Order

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for \$395 or **r-tree** for \$295. (When ordered together, **r-tree** is only \$255). For VISA, MasterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 2606 Johnson Drive, Columbia, MO 65203.

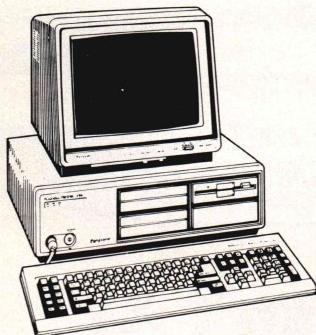


Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp. Unix is a registered trademark of AT&T.

Circle no. 93 on reader service card.

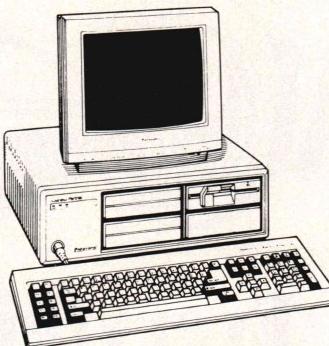
CODE BLUE



PANASONIC FX-800

80286 8MHz • 1.2 MB Floppy
30 MB Hard Disk • 512K RAM
Monitor and Graphics Card

\$2199



PANASONIC FX-600

8086 7.14 MHz • One 360K Floppy
20 MB Hard Disk • 640K
Monitor and Graphics Card

\$1375



TOSHIBA T1100 PLUS

2 720KB Drives
8 hr. Battery • 640K

CALL

Software

Aldebaran Labs

Source Print

Computer Innovations

C-86
Introducing C
C to dBase
CI Probe
CI ROMPac

Metafile Interpreter
Plotting System
Solutions Chart

Logimouse w/PLUS & CAD
Modula-2/86 Compiler

159

Alpha Computer

ACS Time Series

CompuView
Vedit
Vedit Plus

Solutions Plottalk
Solutions Terminal
Impulse Engineering
Fortran Addenda
Fortran Addendum

Modula 2/86
Modula 2/86 PLUS
Modula 2 Library Sources
Modula 2 Make Utility
Modula 2 ROM Package

99
144
84
26
177

For-Winds

Forlib-Plus

Scientific Subroutine Pkg.

Strings & Things

Vitamin C
VC Screen Forms Designer
Custom Sftw Systems
PC/VI

PC/Forth
PC/Forth Plus
Adv. Color Graphics Support
Enhanced Graphics Support

Modula 2 RunTime Debugger
Turbo-Modula Translator
Modula 2 Utilities Pack.
Modula 2 Windows Pack.

57
43
43
43

Arity

Expert System Dev. Pkg.

File Interchange Toolkit

Prolog Compiler/Interpreter

Prolog Interpreter

Screen Design Toolkit

SQL Development Package

Standard Prolog

Combo Package

Data Light
C Compiler
Developer Kit

8087 Support
Interactive Symbolic Debugger
Native Code Optimizer

Luguru
Epsilon
Mansfield Software

157
104
104

Blaise

Asynch Manager-C

Asynch Manager-Pascal

C Tools

C Tools 2

C Tools Plus

EXEC Program Chainer

Pascal Tools

Pascal Tools 2

Pascal Tools & Tools 2

Runoff Text Formatter

Turbo Asynch Plus

Turbo Power Tools Plus

View Manager-C

View Manager-Pascal

Data Base Decisions
Periscope I

Periscope II w/NIM Breakout
Periscope II-X

Kredit
Personal REXX
Mark Williams

287
57
105

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

DeSmet
dBase-C Translator

319
dBase/C Translator (XENIX) CALL

Mark Williams MWC-86
Let's C
Let's C w/csd Source Debugger

57
105

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Digitalk
Methods

Smalltalk/Comm
Smalltalk/V

MGlobal
CCS MUMPS Single User
CCS MUMPS Multi-User

52
369

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

DWB
The PROFILER

Secret Disk
SideTalk

MicroFocus
Cobol Workbench
Level II Cobol

3399
CALL

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

EcoSoft
Eco-C

Text Management Utilities
TopView Toolbasket

MicroFocus
Cobol Workbench
Level II Cobol

215
CALL

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Fifth Generation
FastBack

TopView Toolbasket w/S.C.
Z-80 C Cross Compiler

COGraphics
COMath
Forms-2

165
265
215

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Gimpel
PC Lint

CFORTH

COGraphics
COMath
Forms-2

215
265
215

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Greenleaf
Greenleaf Functions

Logimouse C
Logimouse C Professional

COGraphics
COMath
Forms-2

215
265
215

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Greenleaf Comm Library

Logimouse C Windows

COGraphics
COMath
Forms-2

215
265
215

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Greenleaf Functions

Logimouse C Professional

COGraphics
COMath
Forms-2

215
265
215

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

Greenleaf Comm Library

Logimouse C Windows

COGraphics
COMath
Forms-2

215
265
215

Boehm

Reflex

Reflex Workshop

Reflex & Reflex Workshop

Turbo Database Toolbox

Turbo Editor Toolbox

Turbo Frameworks Toolbox

Turbo Graphix Toolbox

Word Wizard

Turbo Lightning

Turbo Pascal w/8087 & BCD

Turbo Prolog

Turbo Tutor for Pascal

Word Wizard & Turbo Lightning

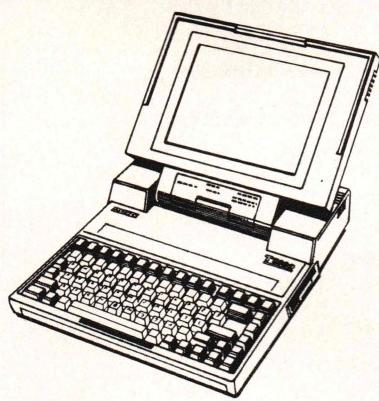
Greenleaf Functions

Logimouse C Professional

COGraphics
COMath
Forms-2

215
265
215

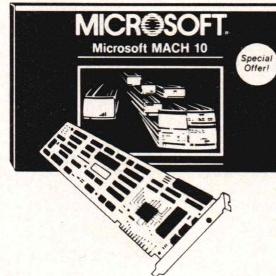
FREE BLUE LABEL SHIPPING*



TOSHIBA 3100

80286 • Gas Plasma Screen
10MB H.D. • 640K
720KB Drive

CALL



MICROSOFT®

MACH 10™.

Performance Enhancement Board
with Windows
and Mouse

\$375

20-Day
Money Back Guarantee!
Call for Details

Orders only:

(800) 232-6442

In California:

(800) 843-2842

Customer service:

(415) 322-0686

Send mail orders to:

Code Blue

508 Waverly Avenue
Palo Alto, CA 94301

Terms & Policies

1. *20-day money back guarantee* on most products. Merchandise must be returned in resalable condition. Call for details.
2. Shipping info: On orders over \$100, we ship free UPS 2nd day air. On orders under \$100, shipping is \$5. Shipping on computers is extra.
3. Prices subject to change without notice.
4. Delivery subject to product availability.
5. P.O.'s accepted from qualified institutions.
6. After 20 days, products can only be returned for repair or replacement. On products not covered by the money back guarantee, they can only be returned for repair or replacement.
7. VISA and MasterCard accepted. COD available at an additional cost.



*On all orders over \$100
to destinations east of the
Rocky Mountains.

| | | | | |
|-------------------------------|-----|---------------------------------|------|--------------------------------|
| Cobol Tools (XENIX) | 299 | Raima | CALL | Software Channels |
| Fortran Compiler | 207 | dbQuery | 155 | ALICE |
| Fortran Compiler (XENIX) | 559 | dbVista single-user | 424 | Software Garden |
| LISP | 159 | dbVista single-user w/S.C. | 424 | Dan Brinklin's Demo Program |
| Macro Assembler | 94 | dbVista multi-user | 839 | Solution Systems |
| Bus Mouse | 119 | dbVista multi-user w/S.C. | 419 | Brief |
| Serial Mouse | 129 | dbVista 1-user w/S.C. (XENIX) | 419 | Spruce Technology |
| Sort | 134 | dbVista multi-user (XENIX) | 419 | FirstTime for Turbo |
| muMath & muSimp | 189 | dbVista multi-user w/SC (XENIX) | 839 | StonyBrook |
| Pascal Compiler | 184 | RDS | 196 | The WATCHER Profiler |
| Pascal Compiler (XENIX) | 429 | C-ISAM | 669 | STSC |
| Tech. Ref. Encyclopedia | 99 | Informix (DOS) | 829 | APL PLUS/PC |
| Windows | 65 | Informix4GL (DOS) | 669 | APL PLUS/PC Spreadsheet |
| Windows Development Kit | 324 | InformixSQL (DOS) | 829 | APL PLUS/PC Tools Vol. 1 |
| Morgan Computing | 124 | Informix (XENIX) | 1259 | APL PLUS/PC Tools Vol. 2 |
| Advance Trace 86 | 124 | Informix4GL (XENIX) | 829 | APL PLUS/UNIX (XENIX) |
| OES Systems | 172 | InformixSQL (XENIX) | 829 | Financial/Stat. Library |
| The Hammer | 172 | Relia | 829 | Pocket APL |
| Opt-Tech Data | 115 | Cobol | 829 | StatGraphics |
| On-Line Help | 115 | Roundhill Computer | 225 | Summit Software |
| Peerless | | Panel | 225 | BetterBasic |
| Scientific Subroutine Library | 135 | Ryan-McFarland | 975 | BetterBasic 8087 Support |
| 50 MORE: Fortran | 96 | RM/Cobol (XENIX) | 589 | BetterBasic Btrieve Interface |
| Phar Lap | 135 | RM/Fortran (XENIX) | 615 | BetterBasic C Interface |
| 386 Debug | 129 | RM/Cobol | 875 | BetterBasic RunTime Module |
| Phoenix | 129 | RM/Cobol 8X ANSI 85 | 379 | Sunny Hill |
| Pasm86 Macro Assembler | 859 | RM/Fortran | 1049 | TurboProfessional |
| Pdisk Hard Disk Utility | 235 | Santa Cruz Operation | 519 | TaskView |
| Pfantasy Pac | 235 | Complete XENIX System | 519 | Toshiba |
| Pfinish Performance Anal. | 138 | XENIX Development System | 149 | 3.5" Drive Kit |
| Pfix-86 Program Debugger | 235 | XENIX Operating System | 479 | True Basic |
| Pfix-86 + Symbolic Debug | 235 | XENIX Test Processing Package | 519 | True Basic w/Converter |
| PforCe C Library | 235 | Lyrix | 685 | True Basic w/Converter/RunTime |
| Plink-86 Overlay Linker | 244 | Networks for XENIX | 215 | Advanced String Library |
| Plink-86 + Enhanced Linker | 319 | SCO Professional | 289 | Asynch Communication Support |
| Pmaker Make Utility | 79 | Scientific Endeavors | 289 | BasicA Converter |
| Pmate Macro Text Editor | 119 | GraphC Mono | 125 | Btrieve Interface |
| Pre-C Lint Utility | 159 | GraphC Color | 335 | Developer's Toolkit |
| Ptel Binary File Transfer | 115 | VTEK | 335 | Formlib |
| Polytron | | Shaw American Tech. | 335 | Hercules Graphics Support |
| PolyBoost | 65 | APT | 62 | Sorting & Searching |
| C Beautifier | 43 | Soft Advances | 85 | RunTime Module |
| C Library I | 73 | DSD86 | 189 | TurboPower Software |
| Power Comm. | 134 | DSD87 | 162 | T-Debug |
| PolyLibrarian | 74 | Softcraft | 119 | Turbo EXTENDER |
| PolyLibrarian II | 110 | Btrieve ISAM Manager | 459 | TurboPower Utilities |
| PolyMake | 74 | Xtrieve Query Utility | 289 | Visual Age |
| PolyXREF-Complete | 175 | Rtrieve Report Generator | 149 | Codesmith-86 |
| PolyXREF-One Language | 105 | Btrieve/N Networks | 65 | Wendin |
| PolyOverlay | 74 | Xtrieve/N Networks | 149 | Operating System Toolbox |
| PVCS Version Control Sys. | 319 | Rtrieve/N Networks | 149 | PCNX Operating System |
| PVMFM Virtual Mem Mgr. | 144 | Software Bottling | 65 | PCVMS |
| R & R Software | | Flash-Up Windows | 92 | XTC Text Editor with Source |
| Janus/ADA C Pack | 85 | Screen Sculptor | 92 | Wizard Systems |
| Janus/ADA D Pack | 779 | | | C Compiler |

Circle no. 370 on reader service card.

354

HELP! PROGRAMMERS

On-Line HELP! Function Library

HELP! is an on-line utility library that gives your software instant, context-sensitive, pop-up HELP! windows at the touch of a key. Link your C code with the HELP! library, tell HELP! which window is current, and HELP! does the rest — fast. HELP! writes directly to video memory with no flicker or snow.

HELP! Runs on PCs and compatibles, B+W or CGA. Source code is available.

HELP! Composer

Compose your HELP! windows interactively. Control HELP! window text, size, colors, position, borders, titles. The HELP! Composer runs as a standalone utility or link it along with your programs to watch each HELP! window take form against the backdrop of your own screen designs. The HELP! Composer builds an ASCII text file to describe all the HELP! windows for each application.

Complete Windows and Pop-Down Menu Library

A bonus: Use the window and pop-down menu functions from the HELP! library in your own programs. This is a complete C window package. Open and close windows, fill windows with text, scroll, select with a cursor bar, promote a window from the background, move a window.

Window Text Editor

Use the full-featured, window-oriented HELP! Text Editor to collect text data into your application. Open a window and call the editor.

HELP! links with programs compiled by most MS-DOS C compilers: Aztec, CI-C86, Datalight, DeSmets, Eco-C88, High C, Lattice, Lets C, Microsoft, Whitesmiths, Wizard. Most memory models are supported. The HELP! distribution package includes libraries, a C demo program, and a complete Programmer's Manual.

HELP! \$49. HELP! with Source Code: \$149. Demo diskette: \$10. deducted from your order. Specify compiler. MasterCard and VISA are accepted. (FL residents add 5% sales tax.)

C SOFTWARE TOOLSET

2983 Newfound Harbor Drive — Merritt Island, FL 32952 — (305) 453-0257

Circle no. 235 on reader service card.

*Does this look familiar?
What if each change
you made to your
program was ready to
test in seconds instead
of minutes?*



"The SLR tools will change the way you write code. I don't use anything else.", Joe Wright

RELOCATING MACRO ASSEMBLERS • Z80 • 8085 • HD64180

- Generates COM, Intel HEX, Microsoft REL, or SLR REL
 - Intel macro facility
 - All M80 pseudo ops
 - Multiple assemblies via command line or indirect command file
 - Alternate user number search
 - ZCPR3 and CP/M Plus error flag support, CP/M 2.2 submit abort
 - Over 30 user configurable options
 - Descriptive error messages
 - XREF and Symbol tables
 - 16 significant characters on labels (even externals)
 - Time and Date in listing
 - Nested conditionals and INCLUDE files
 - Supports math on externals
- requires Z80 CP/M compatible systems with at least 32K TPA

\$49.95

1622 N. Main St., Butler, PA 16001
(412) 282-0864 (800) 833-3061

Circle no. 78 on reader service card.

HASHING FOR SEARCHING

Listing One (Text begins on page 34.)

{Listing One: Declarations of data structures used by hashing and symbol table routines.}

```

CONST
    symbol_hash_size = 100;
    {Number of buckets - 1 in the hash table.
     I believe it should be a prime - 1.}

TYPE
    str255 = String [255]; {General large str}
    symbol_data = RECORD
        {Data to be associated with identifier}
        usecount: INTEGER;
    END;

    symbol_name = String [255];
    {Symbol identifier is a string}

    symbol_ptr = ^symbol_Type;
    symbol_range = 0..symbol_hash_size;
    symbol_Type = RECORD
        {identifier and its data}
        sym_chain: symbol_ptr;
        {Ptr to next symbol in list}
        sym_data: symbol_data;
        {Type declared in the main program}
        sym_name: symbol_name;
        {Symbol name or identifier}
    END;

    symbol_control = RECORD
        {Declare one of these in main program for
         each symbol table to be used}
        symbols, searches, notfound: INTEGER;
        probes: REAL;
        {Real because some counts exceed 32767}
        this_bucket: symbol_range;
        {Bucket # of last referenced symbol}
        this_symbol: symbol_ptr;
        {Pointer to last referenced symbol}
        sym_ptr: ARRAY [symbol_range] OF symbol_ptr;
        {Buckets}
    END;

```

End Listing One

Listing Two

{Listing 2: Routines to initialize the symbol table, insert a symbol, and locate a symbol, without MTF.}

```

FUNCTION symbol_size
    (VAR s_name: symbol_name): INTEGER;
    {Return the size of memory required to contain
     a symbol named in s_name.}

BEGIN
    symbol_size := SIZEOF (symbol_ptr)
                  + SIZEOF (symbol_data)
                  + SUCC (LENGTH (s_name));
END;

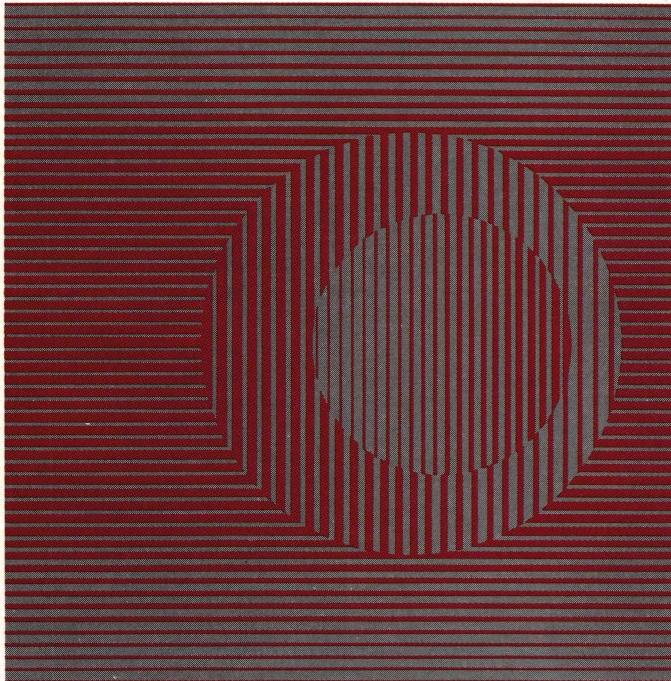
PROCEDURE symbol_init (VAR sym: symbol_control);
    {Initialize symbol control pointers. Call this
     before the first use of a Symbol_Control area.}
    VAR
        i: symbol_range;
    BEGIN
        WITH sym DO BEGIN
            FOR i := 0 TO symbol_hash_size
                DO sym_ptr [i] := NIL;
            this_bucket := 0;
            this_symbol := NIL;
            symbols := 0;
            searches := 0;
            probes := 0.0;
            notfound := 0;
        END;
        END;

        PROCEDURE symbol_put (VAR sym: symbol_control;
                               s_name: symbol_name; VAR s_data: symbol_data);
            {Insert symbol name and data in table. This
             routine does not check for duplicate symbol.}
            BEGIN
                WITH sym DO BEGIN

```

(continued on page 47)

PROFESSIONAL PROLOG FOR THE PROFESSIONAL PROGRAMMER



The AI Starter Kit
from LOGICWARE.
All for \$220.00.

Clip and mail to:

LOGICWARE INC., Micro Division,
70 Walnut Street, Wellesley, MA. 02181

Name _____

Address _____

State/Prov. _____

Zip/Postal Code _____

Please bill my charge card

Charge Card Name: _____

Account #

You are in the market for serious AI tools. You are just getting started — but you have plans for bigger, more serious work. Your purchasing criteria:

- You want an inexpensive product — but not an "AI toy".
- You are doing your initial work on a PC — but you want a product that is flexible and portable. You want the option of expanding your serious work to other, larger computers.
- You do not want to face any dead ends a month or a year down the road.
- You want to invest in a product line that does not leave you stranded when you take that inevitable next step.

Logicware Inc., the leading supplier of Prolog and Prolog-based AI tools, has a solution for you — our AI Starter Kit. This starter kit includes:

- The P-550 product:
An inexpensive, but full-featured implementation of Prolog. It consists of our full MProlog interpreter and our interactive programming environment with a full screen editor. This product also contains *Eagle Graphics* — a full set of graphics predicates that allow you to create three-dimensional graphics for your PC applications.
- The Primer:
Our introduction to the Prolog language and to the concepts of logic programming — a 500-page textbook and two-diskette tutorial software. The best hands-on introduction to Prolog on the market today.

Both P-550 and the Primer software require 512K RAM, DOS 2.0 (or later) and a minimum of two floppy drives.

MProlog is also available for IBM mainframes, DEC VAX and M68000 workstations.

To order, clip and mail the postcard below. For more information call (416) 672-0300 and ask for Chris Glenn.

Circle no. 366 on reader service card.

Logicware Inc.

Company _____

Telephone _____

Enclosed is a check or money order to LOGICWARE INC.

Valid from ____ / ____ to ____ / ____

Signature: _____

CLEAN SCREEN MACHINE

Data Entry

Menus

Windows

Prototyping

Toolkit

C-scape

■ Total Screen Control

C-scape is a combination screen generator and library of screen I/O functions. Written for C programmers, C-scape brings a fresh approach to the need for an easy-to-learn and use, but truly powerful and flexible screen management tool.

C-scape's kernel is your most powerful ally. Without requiring parameters you'll never use, it allows you to create tailored functions with ease and simplicity. Each key is individually definable. If you know `printf()`, you can use C-scape. C-scape's kernel provides a veritable screen design and construction toolkit to rewrite our functions or to write your own.

■ Most Powerful Prototyping Available

C-scape offers a unique approach to prototyping your software. You may use **Dan Bricklin's Demo Program** to create, edit, and view your screens (you can even capture existing screens from other programs), and then use C-scape's **demo2c** utility to convert each screen to code. You can design each screen with attributes such as colors, menu selections, data entry fields (including type, validation, and field naming), masking, and text, and then automatically convert the entire screen to code.

■ Powerful Function Library

Use C-scape's functions for Lotus-like, pull-down, or your own menu designs, automatic scrolling, pop-up windows (number limited only by RAM), logical colors, help, time and date, yes/no, tickertape fields, secure and protected fields, and many others, to turn your demo into a fully functioning and complete program in a fraction of the time spent coding screens from scratch.

C-scape's extensive library includes just about all the data entry and display functions you'll ever need, including money functions, fully definable borders, and orthogonal field movement (get the latest list by calling for more information). And modifying our functions or writing your own is easy. C-scape adjusts automatically for CGA, EGA, monochrome, and the Hercules Graphics Card Plus in RamFont mode, and optionally writes directly to video memory, so it's flexible and fast.

■ Bridges to Power

C-scape includes examples of how to bridge to other powerful tools such as **c-tree** and **db_VISTA**. You'll be integrating demos to dictionaries to file handlers and database managers in no time. You can even use C-scape to provide the screen design for AI applications, using TransLisp Plus and other packages that support calls to C.

■ Clean, Complete Documentation

C-scape's documentation is a clear example of how to write for programmers in a hurry. A short introduction uses helpful examples to explain the C-scape design. Each function is documented separately. An index makes reference easy, and a quick-reference card provides a synopsis of each function.

■ Source Code Included/Portable/No Royalties/No Runtime License

Providing source code at no additional cost gives you the freedom to modify existing functions without raising cost as a barrier. The source code includes all the low level routines you might need to port C-scape to an unsupported machine or compiler. Speaking of barriers, you pay *no royalties or runtime license fees*, either.

■ Toll Free Support and Bulletin Board

To make your job even easier, we provide technical support (toll free), and access to our 24-hour Bulletin Board.

■ Easy Prices/Risk-Free Terms

Try C-scape for 30 days. If you are not satisfied, return it for a refund. When you register, we send you source code. Order C-scape today, or call toll free for more information. See why some very major groups are standardizing on C-scape.

C-scape (Lattice/Microsoft/others) **Only \$179**

C-scape with Dan Bricklin's Demo Program **\$249**

Please add \$3 for shipping; Massachusetts orders add 5% sales tax.



CALL TODAY!
617-491-7311
800-233-3733

Oakland Group, Inc.



675 Massachusetts Avenue, Cambridge, MA 02139-3309

HASHING FOR SEARCHING

Listing Two (*Listing continued, text begins on page 34.*)

```

this_bucket := symbol_hash (s_name);
GETMEM (this_symbol, symbol_size (s_name));
WITH this_symbol^ DO BEGIN
    sym_chain := sym_ptr [this_bucket];
    sym_data := s_data;
    sym_name := s_name;
    sym_ptr [this_bucket] := this_symbol;
END;
symbols := SUCC (symbols);
END;
FUNCTION symbol_get
  (VAR sym: symbol_control; s_name: symbol_name;
   VAR s_data: symbol_data): BOOLEAN;
{Retrieve a symbol. If the symbol is found,
set s_data to the data stored by the last call
to symbol_put specifying the same symbol name,
point this_symbol to the symbol table entry, and
return TRUE. If the symbol is not found leave
s_data unchanged, leave this_symbol undefined,
and return FALSE. This version does NOT
implement the MTF algorithm.}
VAR
  p: symbol_ptr; {work pointer}
BEGIN
  WITH sym DO BEGIN
    this_bucket := symbol_hash (s_name);
    p := sym_ptr [this_bucket];
    symbol_get := FALSE;
    searches := SUCC (searches);
    IF p = NIL THEN
      notfound := SUCC (notfound);
    WHILE p <> NIL DO WITH p^ DO BEGIN
      probes := probes + 1.0;
      IF s_name = sym_name THEN BEGIN
        {found it!}
        s_data := sym_data;
        this_symbol := p;
        p := NIL;
        symbol_get := TRUE;
      END ELSE BEGIN
        {not this one, chain to the next}
        p := sym_chain;
        if p = NIL THEN
          notfound := SUCC (notfound);
      END;
    END;
  END;
END;

```

Listing Three

{Listing 3: Hash functions, presented as a single Pascal function with case statement controlled by a global variable ``hashtype'' to select one of the four routines.}

{First the table used by the CRC-16 routine,
this from a public domain file uncompression
program: DeArc, by Bela Lubkin.}

```

const crctab : array [0..255] of integer =
($0000, $C0C1, $C181, $0140, $C301, $03C0, $0280,
$C241, $C601, $06C0, $0780, $C741, $0500, $05C1,
$C481, $0440, $CC01, $00C0, $0D80, $CD41, $0F00,
$CF01, $C8E1, $0E40, $0A00, $CAC1, $CB81, $0B40,
$C901, $09C0, $0880, $C841, $D801, $18C0, $1980,
$D941, $S1B00, $DBC1, $DA81, $1A40, $1E00, $DEC1,
$DF81, $1F40, $DD01, $1DC0, $1C80, $DC41, $1400,
$D4C1, $D581, $1540, $D701, $17C0, $1680, $D641,
$D201, $12C0, $1380, $D341, $1100, $D1C1, $D081,
$1040, $F001, $30C0, $3180, $F141, $3300, $SF3C1,
$F281, $3240, $3600, $F6C1, $F781, $3740, $F501,
$35C0, $3480, $F441, $3C00, $F7C1, $F8D1, $3D40,
$FF01, $3FC0, $3E80, $FE41, $FA01, $3AC0, $3B80,
$FB41, $3900, $F9C1, $F881, $3840, $2800, $SE8C1,
$EE981, $2940, $SEB01, $2B20, $S2A80, $SEA1, $SEE01,
$2EC0, $2F80, $SEF41, $2D00, $EDC1, $SEC81, $2C40,
$E401, $24C0, $2580, $SE541, $2700, $SE7C1, $SE681,
$2640, $2200, $SE2C1, $SE381, $2340, $SE101, $21C0,
$2080, $SE041, $SA001, $60C0, $6180, $SA141, $6300,
$A3C1, $A281, $6240, $S6600, $A6C1, $A781, $6740,
```

(continued on next page)

**★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★★ IBM BASIC/BASICA ★★**

The **Aspen Systems Subroutine Editor (ASE)** you can call from your programs. Design your own screen layouts -- update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
 - ★ **VERSATILE** - customize input for any application.
 - ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
 - ★ **EASY TO USE** - data and screen layouts described in a single map.
 - ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
 - ★ **AFFORDABLE-**

| | |
|--------------------------------------|------|
| ASE is still only..... | \$99 |
| DEMO (See what ASE can do for you!). | \$ 3 |

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE**, **ASP** is:

**TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE**

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... \$130
ASE and **ASP** include support and demo programs

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Circle no. 277 on reader service card.

C CODE FOR THE PC

source code, of course

| | |
|--------------------------------------|-------|
| GraphiC 3.0 hi-res color plots . . . | \$300 |
| Panache C Program Generator . . . | \$125 |
| QC88 C Compiler | \$90 |
| Concurrent C | \$45 |
| Coder's Prolog in C | \$45 |
| Biggerstaff's System Tools | \$40 |
| Translate Rules to C | \$30 |
| LEX | \$25 |
| YACC & PREP | \$25 |
| tiny-c interpreter & shell | \$20 |
| C Tools | \$15 |

The Austin Code Works

11100 Leafwood Lane

Austin, Texas 78750-3409

(512) 258-0785

Circle no. 250 on reader service card.

Free shipping on prepaid orders

No credit cards

HASHING FOR SEARCHING

Listing Three (*Listing continued, text begins on page 34.*)

```

$A501, $65C0, $6480, $A441, $6C00, $ACC1, $AD81,
$6D40, $AF01, $6FC0, $E680, $AE41, $AA01, $6AC0,
$6B80, $AB41, $6900, $A9C1, $AA81, $6840, $7800,
$B8C1, $SB981, $7940, $BB01, $7BC0, $7A80, $BA41,
$BEC0, $7F80, $BF41, $7D00, $BDC1, $BC81,
$7C40, $B401, $74C0, $7580, $B541, $7700, $B7C1,
$B681, $7640, $7200, $B2C1, $B381, $7340, $B101,
$71C0, $7080, $B041, $5000, $90C1, $9181, $5140,
$9301, $53C0, $5280, $9241, $9601, $56C0, $5780,
$9741, $5500, $95C1, $9481, $5440, $9C01, $5CC0,
$5D80, $9D41, $5F00, $9FC1, $9E81, $5E40, $5A00,
$9AC1, $9B81, $5B40, $9901, $59C0, $5880, $9841,
$8801, $48C0, $4980, $8941, $4B00, $8BC1, $8A81,
$4A40, $4E00, $8EC1, $8FB1, $4F40, $8D01, $4DC0,
$4C80, $8C41, $4400, $84C1, $8581, $4540, $8701,
$47C0, $8460, $8641, $8201, $42C0, $4380, $8341,
$4100, $81C1, $8081, $4040 );

FUNCTION symbol_hash
(VAR s_name: symbol_name): symbol_range;

{Hash the symbol name to a number between
0 and the hash table size.}
VAR
  i, j: INTEGER;
BEGIN
  CASE hashtype OF
    1: BEGIN {Sum of the characters + length}
      j := 0;
      FOR i := 0 TO LENGTH (s_name) DO
        j := j + ORD (s_name [i]);
      symbol_hash :=
        j MOD SUCC (symbol_hash_size);
    END;
    2: BEGIN {First + Last + Length}
      symbol_hash :=
        ((ORD (s_name [1]) SHL 8)
        + ORD (s_name [Length (s_name)])
        + Length (s_name))
        MOD SUCC (symbol_hash_size);
    END;
  END;
END;

```

```

3: BEGIN {HashPJW}
  j := 0;
  FOR i := 1 TO LENGTH (s_name) DO BEGIN
    j := (j SHL 4) + ORD (s_name [i]);
    IF (j AND $F000) <> 0 THEN
      j := j XOR (j SHR 12) AND $0FFF;
  END;
  symbol_hash := (j AND $7FFF)
    MOD SUCC (symbol_hash_size);
END;

4: BEGIN {CRC-16}
  j := 0;
  FOR i := 1 TO LENGTH (s_name) DO
    j := (j SHR 8) XOR
      crctab [(j XOR ORD
        (s_name [i])) AND $00FF];
  symbol_hash := (j AND $7FFF)
    MOD SUCC (symbol_hash_size);
END;

else symbol_hash := 0;
  {Not specified, punish the user}
END;
END;

```

End Listing Three

Listing Four

{Listing 4: Symbol distribution function U(h,t)}

```

FUNCTION symbol_distribution
(VAR sym: symbol_control): REAL;

```

{Compute the distribution test as outlined in
Aho, et al. This function approaches 1.0 as
the "randomness" of the hashing improves.}

```

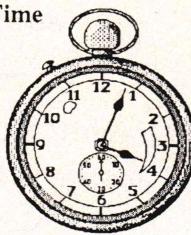
VAR
  p: symbol_ptr;
  b, n, m, r: REAL;
  i: symbol_range;
  j: INTEGER;
BEGIN

```

(continued on page 50)

Along With Your Computer, Your Time
is the Most Important Thing You
Own... So Why Waste It?

*Quilt Programmer Productivity
Tools will help you manage your
software projects and get control
of your time!*



SRMS™

Software Revision Management System

SRMS is a full featured revision control system that allows programmers, system managers, and librarians to get control of software projects. SRMS stores every version of a source program in a single ASCII file, with the ability to restore each and every version easily and quickly. Only changes (deltas) between versions are actually stored, so disk usage is drastically reduced. Each version is stored with a unique Version Identification String and unlimited user supplied comments, making identification and documentation tasks easy. SRMS supports projects written in any language, and allows you to use your current editor and compiler without conflict. Version MERGE facilities help consolidate parallel development efforts. SRMS also incorporates the use of classes to bind various versions of different modules together for reconstructing a particular software system release. DOS directory pathnames and environment variable support also aid you in your work. Full typeset documentation with tutorial and project management chapters.

SRMS \$ 125.00

QMAKE™

Automatic System Generation Utility

QMAKE is a powerful and flexible utility that can be used to control the rebuilding of simple or complex software systems. QMAKE relieves the software developer of the task of remembering which modules of a system need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them. Once you specify the relationships and commands necessary to rebuild your system, and place them in a control file, called a makefile, QMAKE can be used to rebuild your system through the invocation of a single command.

QMAKE will work with most compilers, assemblers, and linkers. It supports full macro definitions, UNIX makefile compatibility, recursive invocations, and interpretation of even the most complex dependencies.

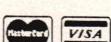
QMAKE will also interface with our Software Revision Management System providing you with a complete set of productivity tools to handle any of your project requirements. Full typeset documentation with tutorial.

QMAKE \$ 99.00



7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and
Dealer Inquiries Welcome



99⁴⁴/₁₀₀ % UNIX EQUIVALENT Power Tools

TOOL BOX 1 \$25

cat cp expand fold more mv newer
page rm unexpand

TOOL BOX 2 \$30

diff fgrep head od see tail tee uniq
wc xcp

SYSMAKE \$25

with advanced features from the
new UNIX Version 8

For MSDOS and CP/M

I/O redirection • wildcard expansion
exclusions • full documentation

NIRVONICS inc

P.O. Box 5062
Plainfield, NJ 07061

(201) 561-2155

Trademarks — UNIX: Bell Labs, MSDOS: Microsoft, CP/M: Digital Research
Circle no. 331 on reader service card.

TEXT LINE 6 COL: 12 FILE: VEDITPLUS.TXT INSERT

WINDOW 1

VEDIT PLUS is an advanced editor that makes your program development and word processing as efficient and easy as possible. VEDIT PLUS is simple enough to learn and use for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. VEDIT PLUS is particularly suited for writing all types of programs and lengthy documents such as reports or manuscripts.

This shows how VEDIT PLUS can perform windowing. One window is used for word processing, a second for program development, and the third for commands.

WINDOW 2

```
bldlist ( infile )
FILE *infile;
{
register i;
struct node *ptr;

for (i=0; i<termlim; i++) {
ptr = malloc ( NODESIZE );
if (!i)
head = tail = ptr;
else
tail->next=ptr;
tail=ptr;
}
tail->next=NULL;
load _str( &(tail->header),
return ( termlim );
}
```

WINDOW \$

| | | | | | | | |
|---------|------|---------|------|--------|------|---------|------|
| VPLUSPC | .COM | INSTALL | .EXE | LIHARD | .BAT | T | .BAT |
| LIGHT | .COM | ENVI | .COM | LONG | .NUM | DISK | .DIC |
| VEDIT | .INI | RAM2 | .DIC | KEYS | IBM | THES | .DIC |
| LIGHT | .HLP | RAM3 | .DIC | PRINT | .EXC | INSTALL | .INI |

VEDIT PLUS

#1 CHOICE IN PROGRAMMABLE EDITORS

VEDIT PLUS has been the #1 choice of professionals since 1980. Our latest release is even better - you can open windows to simultaneously edit several files, access many editing functions with pop-up menus, use keystroke macros to speed editing, and run other programs or DOS commands from within VEDIT PLUS.

Whether your needs are program development, technical writing or word processing, VEDIT PLUS is your answer. With over 40,000 users you can depend on VEDIT PLUS to perform consistently and reliably. It is simple enough to learn for the novice, yet has the speed, flexibility and power to satisfy the most demanding professional.

Compare. No other editor is as powerful - unlimited keystroke macros, instant 'off-the-cuff' command macros utilizing a complete programming language, single command file comparison, special word processing and programming features. No other editor is as easy to use - on-line help, pop-up menus, 75 page tutorial, 380 page manual, and VEDIT PLUS is completely customizable.

Fully supports color windows on IBM CGA & EGA, and even windows on most CRT terminals (including CRT's connected to an IBM PC). Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. Order direct or from your dealer. \$185

"To sum things up, VEDIT PLUS is a small, fast, sophisticated editor with a wealth of features and a good macro language. It offers many rewards for the dedicated programmer."

Computer Language, Chris Wolf, Scott Lewis, Mark Gayman 6/86

"VEDIT PLUS is a wholly remarkable program: blindingly fast, extremely powerful, and highly flexible."

Profiles Magazine, Robert Lavenda 4/86

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. MS-DOS is a registered trademark of Microsoft. CP/M is a registered trademark of Digital Research. WordStar is a registered trademark of MicroPro.

Circle no. 122 on reader service card.

- ### VEDIT PLUS FEATURES
- Simultaneously edit up to 37 files of unlimited size.
 - Split the screen into variable sized windows.
 - 'Virtual' disk buffering simplifies editing of large files.
 - Memory management supports up to 640K.
 - Execute DOS commands or other programs.
 - MS-DOS pathname and CP/M user number support.
 - Horizontal scrolling - edit long lines.
 - Flexible 'cut and paste' with 36 text registers.
 - Customization - determine your own keyboard layout, create your own editing functions, support any screen size, any CRT.
 - Optimized for IBM PC/XT/AT. Also 132 column & up to 70 lines.
- ### EASY TO USE
- Interactive on-line help is user changeable and expandable.
 - On-line integer calculator (also algebraic expressions).
 - Single key search and global or selective replace.
 - Pop-up menus for easy access to many editing functions.
 - Keystroke macros speed editing, 'hot keys' for menu functions.
- ### FOR PROGRAMMERS
- Automatic Indent/Indent for 'C', PL/I or PASCAL.
 - Match/check nested parentheses, i.e. '{' and '}' for 'C'.
 - Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
 - Optional 8080 to 8086 source code translator.
- ### FOR WRITERS
- Word Wrap and paragraph formatting at adjustable margins.
 - Right margin justification.
 - Support foreign, graphic and special characters.
 - Convert WordStar and mainframe files.
 - Print any portion of file; separate printer margins.
- ### MACRO PROGRAMMING LANGUAGE
- 'If-then-else', looping, testing, branching, user prompts keyboard input, 17 bit algebraic expressions, variables.
 - CRT emulation within windows, Forms entry.
 - Simplifies complex text processing, formatting, conversions and translations.
 - Complete TECO capability.
 - Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Main menu

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

HASHING FOR SEARCHING

(Listing continued, text begins on page 34.)

```

r := 0.0;
WITH sym DO BEGIN
  FOR i := 0 to symbol_hash_size DO BEGIN
    p := sym_ptr [i];
    j := 0;
    WHILE p <> NIL DO
      WITH p^ DO BEGIN {count the list}
        p := sym_chain;
        j := SUCC (j);
      END;
    b := j;
    r := r + (b * (b + 1.0)) / 2.0;
  END;
  m := SUCC (symbol_hash_size);
  n := symbols;
  symbol_distribution := r /
    ((n / (2.0 * m)) * (n + 2.0 * m - 1.0));
END;

```

End Listing Four

Listing Five

(Listing 5: Symbol search with MTF)

{Note: for the purposes of the test program, the application of MTF is controlled by a global boolean variable ``mtf'' set by the main program. The test for this boolean should be removed in a production version of the routine. MTF with all its performance advantages is accomplished with the addition of seven lines of code!}

```

FUNCTION symbol_get
  (VAR sym: symbol_control; s_name: symbol_name;
   VAR s_data: symbol_data): BOOLEAN;

```

{Retrieve a symbol. If the found, s_data is set to the data stored by the last call to symbol_put specifying the same symbol name, this_symbol points to the symbol table found, the symbol is moved to the front of the chain, and the function

returns TRUE. If the symbol is not found s_data is unchanged, this_symbol is undefined, and the function returns FALSE.}

```

VAR
  p: symbol_ptr; {work pointer}
BEGIN
  WITH sym DO BEGIN
    this_bucket := symbol_hash (s_name);
    p := sym_ptr [this_bucket];
    symbol_get := FALSE;
    this_symbol := NIL;
    searches := SUCC (searches);
    IF p = NIL THEN notfound := SUCC (notfound);
    WHILE p <> NIL DO WITH p^ DO BEGIN
      probes := probes + 1.0;
      IF s_name = sym_name THEN BEGIN
        {found it!}
        IF this_symbol <> NIL THEN IF mtf THEN
          BEGIN {Move it to the front}
            this_symbol^.sym_chain := sym_chain;
            sym_chain := sym_ptr [this_bucket];
            sym_ptr [this_bucket] := p;
          END;
        s_data := sym_data;
        this_symbol := p;
        p := NIL;
        symbol_get := TRUE;
      END ELSE BEGIN
        {not this one, chain to the next}
        this_symbol := p;
        p := sym_chain;
        if p = NIL THEN
          notfound := SUCC (notfound);
      END;
    END;
  END;
END;

```

End Listings

WINDOWS FOR DATA™

DATA ENTRY WINDOWS MENUS HELP

Windows for Data does the hard jobs that others can't — we **guarantee** it. Makes standard display and entry tasks easy. Reliable. Compact. Portable.

DATA ENTRY: The most complete and flexible data entry system on the market. Pop-up data-entry windows; field types for all C data types, plus decimals, dates, and times; auto conversion to and from strings for all field types; system and user-supplied validation functions; range-checking; scrollable context-sensitive help; required and must-fill fields; programmer-definable edit keys, field types, and field masks. Read field by field or auto-read all fields. Branch and nest window forms. **Virtually every capability of WFD can be modified to meet special needs.**

WINDOWS: WFD is built upon and includes **Windows for C**, the windowing system rated #1 in PC Tech Journal (William Hunt, July 1985). WFD now has more features than ever, including automatic full compatibility with Microsoft Windows and TopView.

UNPRECEDENTED FLEXIBILITY



As many possibilities as Vermont in June.

MENUS: Build multi-level menus in the format of Lotus 1-2-3, Macintosh, or a style of your own choosing.

HELP: Build context-sensitive or menu-driven help systems. Display text in pop-up, scrollable windows.

UNIX, DOS, OR BOTH

WFC and WFD provide source code compatibility between PCDOS and UNIX.

OUR CHALLENGE AND GUARANTEE

If you have an application where no other tool can do the job, try **Windows for Data**. If it doesn't help you solve your problem, RETURN FOR A FULL REFUND. YOU MUST BE SATISFIED.

WINDOWS FOR DATA WINDOWS FOR C

| | | |
|-----------|-------|-------|
| PC DOS* | \$295 | \$195 |
| XENIX-286 | \$595 | \$395 |
| UNIX | CALL | CALL |

Call for **FREE Demo diskette**.

*All popular C compilers; no royalties.

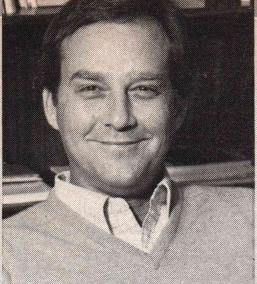


**Vermont
Creative
Software**

21 Elm Ave.
Richford, VT 05476
802-848-7738,
ext. 31.

MasterCard & Visa Accepted. Shipping \$3.50
VT residents add 4% tax.

Circle no. 157 on reader service card.



DR. JACK PURDUM

Get the proven product from the man who wrote the books on "C".

C Programming Guide

After reading the 1st edition, Jerry Pournelle (BYTE Magazine) said: "I recommend this book... Read it before trying to tackle Kernighan and Ritchie." The second edition expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.

\$20

C Self-Study Guide

(Purdum, Que Corp.) Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.

\$17

C Programmer's Library

(Purdum, Leslie, Stegemoller, Que Corp.) This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

\$22

CED Program Editor

CED now supports on-line function help. If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look-alikes.

\$29.95

C Source for Standard Library

Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendental and functions written in assembler.

\$10 (\$20 if not with order)

Developer's Library

Contains all the source code for the standard library, including transcendental and assembler functions. Available with compiler purchase only.

\$25 (\$50 if not with order)

Ecosoft Inc.

6413 N. College Ave.
Indianapolis, IN 46220

THE FIRST PROFESSIONAL 'C' COMPILER FOR UNDER **\$60.00**

Limited Time Offer
FREE
CED TEXT EDITOR WITH
"BUILT-IN" FUNCTION HELP.

Circle no. 89 on reader service card.

NOTHING IN THIS PRICE RANGE EVEN COMES CLOSE.

"Ecosoft's Eco-C is going to turn some major heads."

"Eco-C is the first compiler reviewed that has clearly begun implementing the coming ANSI standard. Eco-C supports prototyping."

"Eco-C performed well on all the benchmarks, generating code that was quite comparable to that of compilers 10 times as costly."

from:
Christopher Skelly
Computer Language,
Feb., 1986

"The driver program is another strength: it compiles and links a list of files and provides a simple MAKE capability."

"This compiler does handle syntax errors much better than average—no avalanche of spurious messages here."

from:
William Hunt
PC Tech Journal,
Jan., 1986

"Eco-C88 is a high-quality package... comparable to systems costing much more."

"Eco-C88 is one of the fastest..."

"It is convenient to use, works well, and produces acceptably compact and fast programs."

from:
Dr. David Clark
Byte, Jan., 1986

Minimum System Requirements:

To use Eco-C88 Release 3.0, you must have:

1. An IBM PC, XT, or AT-compatible computer with monitor.
2. 256K or more memory.
3. Two 360K disk drives, or a hard disk.
4. PC or MSDOS 2.1 or later to include the MSDOS linker.

Eco-C88, mini-make, memfiles and CED are trademarks of Ecosoft Inc. IBM is a trademark of International Business Machines. UNIX is a trademark of Bell Labs. MSDOS and MASM are trademarks of Microsoft.

Full-featured C compiler. Supports all C features, data types (except bit fields), and operators.

New Language Enhancements. You also get prototyping, enum and void data types, plus structure passing and assignment.

Tiered Error Checking. All syntax errors are automatically flagged, but you can select the level of "link-like" semantic error checking you want.

Complete Standard Library. Over 200 functions, many of which are System V compatible for greater source code portability.

Screen and Memory Functions. Now you can write programs that use color, cursor addressing, even ones that let you design your own graphics functions. You also get memfiles™ that allow you to access memory outside the normal data segment as a file.

8087 and 80287 Support. If you have one of these math chips, your programs will take immediate advantage of it. If you don't have one, the code automatically switches to software floating point.

Full Screen Editor. The CED editor is a full screen editor with multiple windows, macro commands, on-line function help, plus a full set of editing commands. (Requires a true IBM PC compatible.) You can edit, compile, link, and execute programs from the editor which greatly reduces development time.

Includes a cc and mini-make™. The UNIX-like cc makes compiling programs a snap. You can run cc from within the CED editor.

ASM or OBJ Output. You can select assembler or relocatable output from the compiler. Both are MASM compatible and ready for use with the MSDOS linker to produce EXE files.

| | Eco-C88 | Lattice | Computer Inn. C86 | Microsoft | Mark Williams |
|--------|-----------|-----------|-------------------|------------|---------------|
| sieve | 12 | 11 | 13 | 11 | 12 |
| fib | 43 | 58 | 46 | 109 | — |
| deref | 14 | 13 | — | 10 | 11 |
| matrix | 22 | 29 | 27 | 28 | 29 |

Computer Language, Feb., 1985, p. 79. Reproduced with permission.

Orders only: **1-800-952-0472**

ORDER FORM CLIP & MAIL TO: Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

- C Compiler \$59.95 _____
 C Programming Guide \$20.00 _____
 C Self-Study Guide \$17.00 _____
 C Programmer's Library \$22.00 _____
 CED Program Editor \$29.95 _____
 C Source for Standard Library \$10.00 (\$20.00 if not with order) _____
 Developer's Library \$25.00 (\$50.00 if not with order) _____

SHIPPING \$4.00

TOTAL (IND. RES. ADD 5% TAX)

PAYMENT: VISA MC AE CHECK

CARD # _____ EXPIR. DATE _____

NAME _____

ADDRESS _____

CITY _____ STATE _____

ZIP _____ PHONE _____

ECOSOFT

Circle no. 89 on reader service card.

Listing One (*Text begins on page 90.*)

```

1 char    *cpy( dest, src )
2 register char   *dest, *src;
3 {
4     /* Works like strcpy but returns a pointer
5      * to the new end of string (ie. to the null).
6      */
7
8     while( *src )
9         *dest++ = *src++ ;
10
11    *dest = 0;
12    return dest;
13 }
```

End Listing One**Listing Two**

```

1 /*          DATE.C  Get the date from dos
2 */
3
4 #include <dos.h>
5
6 extern int      intdos( union REGS*, union REGS* );
7
8 /*****-----*/
9
10 date( month, day, year, day_of_the_week )
11 int      *month, *day, *year, *day_of_the_week ;
12 {
13     /*  Return the month, day, year, and day of the
14      *  week (0 = sunday, 6 = saturday).
15      */
16
17     union REGS      regs;
18
19     regs.h.ah = 0x2a ;
20
21     intdos( &regs, &regs );
22
23     *month          = regs.h.dh ;
24     *day             = regs.h.dl ;
25     *year            = regs.x.cx ;
26     *day_of_the_week = regs.h.al ;
27 }
28
29 /*****-----*/
30
31 #ifdef DEBUG
32
33 main()
34 {
35     int      month, day, year, day_of_the_week;
36
37     date( &month, &day, &year, &day_of_the_week );
38
39     printf("date is %d/%d/%d,      day of the week = %d\n",
40           month, day, year, day_of_the_week );
41 }
42
43 #endif
```

End Listing Two**Listing Three**

```

1 /* HASH.H      Header required by the hash functions in hash.c */
2
3
4 #define MAXNAME 32
5
6 typedef struct element_
7 {
8     struct element_  *next;
9     struct element_ **prev;
10    char    sname[ MAXNAME + 1];
11 }
12 BUCKET;
13
14 typedef struct hash_tab_
15 {
16    BUCKET  **table;      /* Pointer to hash table           */
17    int      size;        /* Max number of elements in table */
18    int      numssyms;    /* number of elements currently in table */
19 }
20 HASH_TAB;
21
```

```

22 /* symname() extracts the name field from a BUCKET:
23 *      p is a pointer returned from findsym, evaluates to the
24 *      contents of the sname field.
25 */
26
27 #define symname(p) ( ((BUCKET*)(p) - 1)->sname )
28
29 extern char    *addsym  (HASH_TAB *, char   *, int      );
30 extern void    delsym  (HASH_TAB *, BUCKET *,        );
31 extern char    *findsym (HASH_TAB *, char   *        );
32 extern HASH_TAB *maketab (unsigned int      );
33 extern void    pstats   (HASH_TAB *        );
34 extern int     ptab     (HASH_TAB *, void (*)());
                                         ); End Listing Three

```

Listing Four

```

1 #include <stdio.h>
2 #include <ctype.h>
3 #include <hash.h>
4
5 /*      HASHTAB.C      General-purpose hash table functions.
6 *
7 *      (C) 1986, Allen I. Holub. All rights reserved.
8 *
9 * The hash table structures are defined in /include/hash.h. A HASH_TAB
10 * is a structure that contains the table size, the number of elements in
11 * the table and a pointer to the table itself, this last an array of
12 * BUCKET pointers. Collisions are resolved by putting the BUCKETS into
13 * a doubly linked list:
14 *
15 *      +-----+ +-----+
16 *      V       | V       |
17 *      +-----+ +-----+ +-----+
18 *      | *----->| | * | *----->| | * | 0 |
19 *      +-----+ +-----+ +-----+
20 *      +-----+      name prev next      name prev next
21 *      +-----+
22 *
23 * The leftmost box is the array pointed at by the HASH_TAB structure.
24 * It's an array of pointers to BUCKETS. The other boxes are BUCKETS. The
25 * "next" field points at the next bucket in the chain or is NULL if there
26 * isn't another bucket. The "prev" field points at the "next" field of the
27 * previous bucket. In the case of the leftmost bucket, it will point at
28 * the actual hash-table element.
29 *
30 * The BUCKET itself is actually a header, similar to the one used by
31 * malloc():
32 *
33 *      +-----+
34 *      | BUCKET |
35 *      +-----+
36 *      | user   | <-- pointer returned from addsym() and findsym()
37 *      = memory =
38 *      |         |
39 *      +-----+
40 *
41 * The pointer returned by addsym() of findsym() can be used as a
42 * structure pointer by the applications program.
43 */
44
45 #define max(a,b)      ((a) > (b) ? (a) : (b))
46 #define min(a,b)      ((a) < (b) ? (a) : (b))
47
48 #define WLEN   ( sizeof(unsigned)*8 ) /* # of bits in an unsigned int */
49 #define MAXINT ((unsigned)-0) >> 1  /* Largest signed integer */
50 #define MAXLEN 128                /* Used by pstat(), max number */
51                                /* of expected chain lengths. */
52
53 *-----
54 * Prototypes for static functions (the global function prototypes are
55 * in hashtable.h):
56 */
57
58 static unsigned hash   (char*,      HASH_TAB* );
59 static int      symcmp (BUCKET**,   BUCKET** );
60
61 *-----
62
63 static unsigned hash( name, tabp )
64 register char   *name;
65 HASH_TAB       *tabp;
66 {
67     /* Compute hash value. Note that the MOD table-length is
68     * done my the calling routine.
69     */

```

(continued on next page)

Fortran Support for IBM PC/XT/AT & Compatibles

Versions Available For:

Microsoft, Supersoft, RyanMcFarland, IBM Professional, Lahey, & IBM Fortran.

Forlib-Plus \$69.95

Supports graphics, interrupt driven communication, program chaining, and file handling/disk support. A Fortran coded subroutine is included which will plot data on the screen either in linear/linear, log/linear, linear/log, or log/log on the appropriate grid.

Strings & Things \$69.95

Supports string manipulations, command line usage, DOS call capabilities, SHELL generation and data transmission, BATCH file control, music generation, PEEKS and POKEs, PORT access, and general register manipulations.

For-Winds \$89.95

Gives the Fortran programmer the capability of generating up to 255 windows on the screen. Each window can be individually scrolled, moved, sized, generated, and removed. Both color and monochrome type displays are supported. Full source code is supplied for customization.

ACS Time Series \$495.00

This is a COMPLETE time series analysis package which contains VERY HIGH SPEED FFTs, Filter generations, convolutions, transfer function calculations, auto and cross spectra calculations, Cepstrum, curve fitting algorithms, coherence calculations, and many other associated routines. The price includes FULL source code.

Fortran Scientific Subroutine Package \$295.00

There are approximately 100 Fortran subroutines included which fall under the following 12 categories:

- 1) Matrix storage and Operations
 - 2) Correlation and Regression,
 - 3) Design Analysis (ANOVA),
 - 4) Discriminant Analysis,
 - 5) Factor Analysis,
 - 6) Eigen Analysis,
 - 7) Time Series,
 - 8) Nonparametric Statistics,
 - 9) Distribution Functions,
 - 10) Linear Analysis,
 - 11) Polynomial Solutions,
 - 12) Data Screening.
- Full source code is included.



ALPHA COMPUTER SERVICE
5300 ORANGE AVENUE SUITE 108
CYPRESS, CALIFORNIA 90630
(714) 828-0286

California Residents
 Include 6% Sales Tax There are NO license fees

Circle no. 321 on reader service card.

Software Developers: FAR EAST BUSINESS OPPORTUNITY

Kanematsu-Gosho, a prominent Japanese trading company, in conjunction with Tokai Create, one of Japan's largest software marketing firms are soliciting submissions of business related applications software for consideration for export to the Japanese market.

The Japanese PC market as presently approximately 2.5 million units with an anticipated annual growth rate of 25% over the next five years. This offer presents a unique opportunity for U.S. software developers to enter this burgeoning market which has previously been difficult to break into.

Your Japanese partners will be taking care of all the translation, marketing and sales functions in this most interesting market. Successful development companies will be rewarded with a substantial revenue stream for the term of the contract with Japan.

Submitted products will be subjected to a series of evaluations. The initial U.S. based screening will be conducted by **CSSL, Inc.** the U.S. representative of the Japanese principals. Successful products will be forwarded to the U.S. offices of Kanematsu-Gosho for further testing and evaluation. Final evaluation will be completed by the parent companies in Tokyo.

Please submit full working versions of your applications software no later than **February 28, 1987** to:

CSSL, INC.
909 Electric Avenue
Seal Beach, CA 90740
Attn: Frank Westall, chairman
Telephone inquiries: 213-493-2471

**ALL SUBMISSIONS WILL BE HELD IN THE
STRICTEST CONFIDENCE.**

C CHEST

Listing Four (*Listing continued, text begins on page 90.*)

```
70     register unsigned h, q;
71
72     for( h = 0; *name ; h += *name++ )
73     ;
74
75
76     return( h % tabp->size );
77 }
78
79 /*-----*/
80
81
82 HASH_TAB      *maketab( maxsyms )
83 unsigned       maxsyms;
84 {
85     /*      Make a hash table of the indicated size. It's a good
86     *      idea to make maxsyms a prime number (though that's not
87     *      required). Some useful primes are:
88     *      47 61 89 113 127 157 193 211 257 293 359 401
89     */
90
91     extern HASH_TAB *calloc();
92     HASH_TAB        *p;
93
94     if( p = calloc(1,(maxsyms * sizeof(BUCKET)) + sizeof(HASH_TAB)))
95     {
96         p->table = (BUCKET **)( p + 1 );
97         p->size = maxsyms ;
98         p->nunsyms = 0 ;
99     }
100    else
101    {
102        err("Insufficient memory for symbol table\n");
103        exit( 1 );
104    }
105
106    return p;
107 }
108
109 /*-----*/
110
111 char   *addsym( tabp, name, size )
112 HASH_TAB      *tabp;
113 char   *name;
114 {
115     /*      Add a symbol to the hash table.
116     */
117
118     BUCKET **p, *tmp ;
119     BUCKET *sym;
120
121     if( !(sym = (BUCKET *) calloc( size + sizeof(BUCKET), 1)) )
122         ferr("Can't get memory for symbol\n");
123
124     strncpy( sym->sname, name, MAXNAME );
125
126     p = &(tabp->table)[ hash(name,tabp) ];
127
128     tmp      = *p ;
129     *p      = sym ;
130     sym->prev = p ;
131     sym->next = tmp ;
132
133     if( tmp )
134         tmp->prev = &sym->next ;
135
136     tabp->nunsyms++;
137     return (char*)(sym + 1);
138 }
139
140 /*-----*/
141
142 char   *findsym( tabp, name )
143 HASH_TAB      *tabp;
144 char   *name;
145 {
146     /*      Return a pointer to the hash table element having the
147     *      indicated name or NULL if the name isn't in the table.
148     *      If more than one such entry is in the table, the most-
149     *      recently added one is found.
150     */
151
152     BUCKET *p ;
```

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN FOR DEC. 21, 1986

| OUTSIDE OKLAHOMA: NO SALES TAX | | | |
|--------------------------------|---------|---------|----------|
| DYNAMIC RAM | | | |
| 1Mbit | 1000Kx1 | 100 ns | \$38.50 |
| 51258 | *256Kx1 | 100 ns | 9.95 |
| 4464 | 64Kx4 | 150 ns | 3.29 |
| 41256 | 256Kx1 | 100 ns | 3.84 |
| 41256 | 256Kx1 | 120 ns | 2.69 |
| 41256 | 256Kx1 | 150 ns | 2.49 |
| 4164 | 64Kx1 | 150 ns | 1.30 |
| EPROM | | | |
| 27512 | 64Kx8 | 200 ns | \$14.95 |
| 27C256 | 32Kx8 | 250 ns | 5.51 |
| 27256 | 32Kx8 | 250 ns | 4.96 |
| 27128 | 16Kx8 | 250 ns | 3.77 |
| 27C64 | 8Kx8 | 200 ns | 4.52 |
| 2764 | 8Kx8 | 250 ns | 3.27 |
| STATIC RAM | | | |
| 62256LP-1232Kx8 | 120 ns | \$14.95 | 5 MHz |
| 6264LP-15 | 8Kx8 | 150 ns | 2.95 |
| 8087 | | | \$112.00 |
| 80287 | | | \$270.00 |
| 8087 | | | 8MHz |
| 80287 | | | 8MHz |

OPEN 6½ DAYS, 7 AM-10 PM: SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY:
Th: Std \$6.40 Fr: P-One \$13.20 BEGGS, OK 74421

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts uP®

MICROPROCESSORS UNLIMITED, INC.

24,000 S. Peoria Ave. (918) 267-4961

No minimum order

Please call for current prices because prices are subject to change. Shipping & insurance extra.
Cash discount prices shown. Orders received by 9 PM CST can usually be delivered to you the next morning, via Federal Express Standard Air for \$6.00, or Priority One for \$13.00!

Circle no. 105 on reader service card.

**DISK FORMAT
CONVERSION**

PC-DOS program
lets your PC
Read/Write/Format
over 300 formats

XENOCOPY-PC™

Fred Cisin

\$79.95 + \$5.00 S/H Sales Tax if CA.

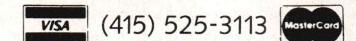
Upgrades available from previous versions

Ask about FREE CP/M emulator! ↗

To Order Contact:

XENOSOFT™

1454 Sixth Street, Berkeley, CA 94710



Circle no. 225 on reader service card.

**Enhance Your Turbo
Pascal™ Programming
with**

TURBO-JET

- Ultra Fast Screen Read and Display
- Advanced String and Numeric Formatting
- Advanced File and Keyboard Handling
- Subdirectory Utilities
- Over 100 Files Included!
- Pascal Source Code included with all Routines
- Routines Crafted in Assembly Language
- No Royalties for Program Use
- Give Programs a Professional Look
- Don't Pay More For Less!
- Dealer Inquiries Invited

TURBO-JET, Only \$39.95

Add \$3.00 for Postage & Handling

NY Residents add sales tax

TOC Business Solutions, Inc.

P.O. Box 129

Old Westbury, N.Y. 11568

MC/VISA (516) 795-2800

Circle no. 345 on reader service card.

640 Kayle MOTHERBOARD KITS: Zenith 150,
IBM PC/XT Compaq Portable & Plus; hp Vectra

1Mbit 1000Kx1 100 ns \$38.50

51258 *256Kx1 100 ns 9.95

4464 64Kx4 150 ns 3.29

41256 256Kx1 100 ns 3.84

41256 256Kx1 120 ns 2.69

41256 256Kx1 150 ns 2.49

4164 64Kx1 150 ns 1.30

EPROM

27512 64Kx8 200 ns \$14.95

27C256 32Kx8 250 ns 5.51

27256 32Kx8 250 ns 4.96

27128 16Kx8 250 ns 3.77

27C64 8Kx8 200 ns 4.52

2764 8Kx8 250 ns 3.27

STATIC RAM

62256LP-1232Kx8 120 ns \$14.95

6264LP-15 8Kx8 150 ns 2.95

8087 5 MHz \$112.00

80287 8 MHz \$270.00

8087 8MHz \$270.00

80287 8MHz \$270.00

8087 8MHz \$270.00

DAN BRICKLIN'S DEMO PROGRAM

Read what they're saying about this new concept in prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."

— PC Magazine, 4/29/86

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."

— Soft-letter, 4/20/86

"Its low price, superb performance, and range of applications practically guarantee that it will be widely used. Four Floppy Rating (8.0)"

— InfoWorld, 3/31/86

"Apparently has a hit on its hands with... a development tool for personal computer software that has won rave reviews from early users."

— Computerworld, 4/7/86

"A gem."

— PC Week, 3/18/86

Product of the Month

— PC Tech Journal, 3/86

ORDER NOW!

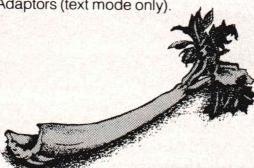
Thousands of developers are designing better products faster and producing more effective demonstrations using Dan Bricklin's Demo Program. You can, too. Act now!



ONLY \$74.95
617-332-2240

Massachusetts residents add \$3.75. Outside of the U.S.A. add \$15.00.

Requires 256k IBM PC/compatible, DOS 2.0 or later. Supports Monochrome, Color/graphics, and EGA Adaptors (text mode only).



SOFTWARE GARDEN

Dept. D

P.O. Box 373, Newton Highlands, MA 02161

Circle no. 314 on reader service card.

C CHEST

Listing Four (Listing continued, text begins on page 90.)

```

238     free( outtab );
239 }
240
241 /*-----*/
242
243 void      pstats( tabp )
244 HASH_TAB *tabp;
245 {
246     /*
247     * Print out various statistics showing the lengths of the
248     * chains (number of collisions) along with the mean depth
249     * of non-empty chains, standard deviation, etc.
250
251     BUCKET *p;           /* Pointer to current hash element */
252     int i;               /* counter */
253     int chain_len;       /* length of current collision chain */
254     int maxlen = 0;       /* maximum chain length */
255     int minlen = MAXINT; /* minimum chain length */
256     int lengths[ MAXLEN ]; /* indexed by chain length, holds */
257                                         /* the # of chains of that length. */
258     int longer = 0;       /* # of chains longer than MAXLEN */
259
260     memset( lengths, 0, sizeof(lengths) );
261
262     for( i = 0; i < tabp->size ; i++ )
263     {
264         chain_len = 0;
265         for( p = tabp->table[i] ; p ; p = p->next )
266             chain_len++;
267
268         if( chain_len >= MAXLEN )
269             ++longer;
270         else
271             ++lengths[chain_len];
272
273         minlen = min( minlen, chain_len );
274         maxlen = max( maxlen, chain_len );
275
276         newsample( chain_len );
277     }
278
279     printf("%d entries in %d element hash table, ",
280           tabp->numsyms, tabp->size );
281     printf("%d (%1.0f%%) empty.\n",
282           lengths[0], ((double)lengths[0]/tabp->size) * 100.00);
283
284     printf("Mean chain length: %d, max=%d, min=%d, deviation=%d\n",
285           running_mean(), maxlen, minlen, deviation() );
286
287     for( i = 0; i < MAXLEN; i++ )
288         if( lengths[i] )
289             printf("%3d chains of length %d\n", lengths[i], i );
290     if( longer )
291         printf("%3d chains of length %d or longer\n", longer, MAXLEN);
292 }
293
294 /*-----*/
295 #ifdef DEBUG
296
297 dptab( addr )
298 HASH_TAB *addr;
299 {
300     BUCKET **p, *bukp ;
301     int i;
302
303     printf("HASH_TAB at 0x%04x (%d element table, %d symbols)\n",
304           addr, addr->size, addr->numsyms );
305
306     for( p = addr->table, i = 0 ; i < addr->size ; ++p, ++i )
307     {
308         if( !*p )
309             continue;
310
311         printf("Htab[%3d] 0x%04x:", i, p );
312
313         for( bukp = *p; bukp; bukp=bukp->next )
314         {
315             printf("= 0x%x (%s) p=0x%x, n=0x%x, user=0x%x\n",
316                   bukp, bukp->sname, bukp->prev, bukp->next, bukp+1);
317
318         }
319     }
320
321     printf(" ");
322 }

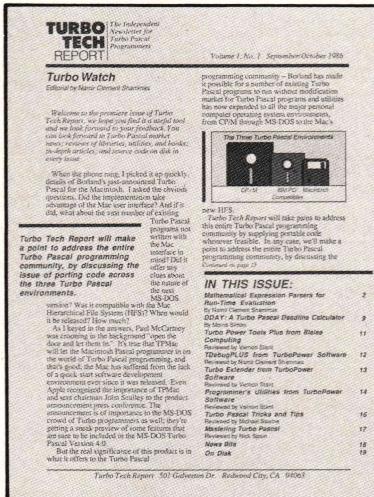
```

(continued on page 58)

Turbo Tech Report Speaks Your Language.

Turbo Pascal
Articles and
Reviews

News and
commentary



A disk filled
with Turbo Pascal
code!

The newsletter/disk publication for Turbo Pascal® users

Are you a devoted Turbo Pascal programmer, tired of reading about other languages? Are you looking for powerful utilities written in Turbo Pascal that you can use to develop software or incorporate into your programs? Are you interested in improving and expanding your Turbo Pascal programming skills?

Then you deserve a subscription to *Turbo Tech Report*, the bimonthly newsletter/disk publication from the publishers of *Dr. Dobb's Journal* and *Micro/Systems Journal*. Each issue delivers more than 250K of Turbo Pascal source code programs on disk, and 20+ pages of articles, Turbo Pascal software and book reviews, and analysis and commentary. It's the only publication delivering such focused technical articles with code on disk—and it doesn't waste your time with information about other programming languages. Each valuable issue contains:

- **Articles** on topics like speedy 3D graphics, mathematical expression parsers, creating global gotos, memory resident and AI applications and more—all written by Turbo experts.
- **Reviews** of the latest Turbo Pascal software programs from companies like Borland

International, Blaise Computing, Media Cybernetics, Nostradamus, TurboPower Software, and more!

• **News and commentary** detailing the latest products and developments in the Turbo Pascal programming community.

• A disk filled with Turbo Pascal code!

You'll get the Turbo Pascal utilities and routines discussed in the newsletter's articles, as well as applications developed by Turbo users from around the world. You'll receive programs that make labels, generate menus, provide faster screen access, transfer files between CP/M and MS-DOS computers, and more!

If you're an expert Turbo Pascal programmer or a novice interested in expanding your Turbo skills, you need a publication that speaks your language: *Turbo Tech Report*. Subscribe today at the special price of just \$99—that's 33% off the regular price of \$150. To order by credit card, call toll-free 1-800-528-6050 ext. 4001 and ask for item 300. Or mail the attached coupon with your payment to *Turbo Tech Report*, 501 Galveston Drive, Redwood City, CA 94063.

Turbo Pascal is a trademark of Borland International Inc.

Circle no. 119 on reader service card.

**COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES**

HS/FORTH

Why HS/FORTH? Not for speed alone, although it is twice as fast as other full memory Forths, with near assembly language performance when optimized. Not even because it gives MANY more functions per byte than any other Forth. Not because you can run all DOS commands plus COM and EXE programs from within HS/FORTH. Not because you can single step, trace, decompile & disassemble. Not for the complete syntax checking 8086/8087/80186 assembler & optimizer. Nor for the fast 9 digit software floating point or lightning 18 digit 8087 math pack. Not for the half megabyte LINEAR address space for quick access arrays. Not for complete music, sound effects & graphics support. Nor the efficient string functions. Not for unrivaled disk flexibility — including traditional Forth screens (sectored or in files) or free format files, all with full screen editors. Not even because I/O is as easy, but far more powerful, than even Basic. Just redirect the character input and/or output stream anywhere — display, keyboard, printer or com port, file, or even a memory buffer. You could even transfer control of your entire computer to a terminal thousands of miles away with a simple >COM <COM pair. Even though a few of these reasons might be sufficient, the real reason is that we don't avoid the objections to Forth — WE ELIMINATE THEM!

Public domain products may be cheap; but your time isn't. Don't shortchange yourself. Use the best. Use it now!

HS/FORTH, complete system: \$395. with "FORTH: A Text & Reference" by Kelly and Spies, Prentice-Hall and "The HS/FORTH Supplement" by Kelly and Callahan

 Visa Mastercard 

HARVARD SOFTWARES

PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

Circle no. 132 on reader service card.

C CHEST

Listing Four (*Listing continued, text begins on page 90.*)

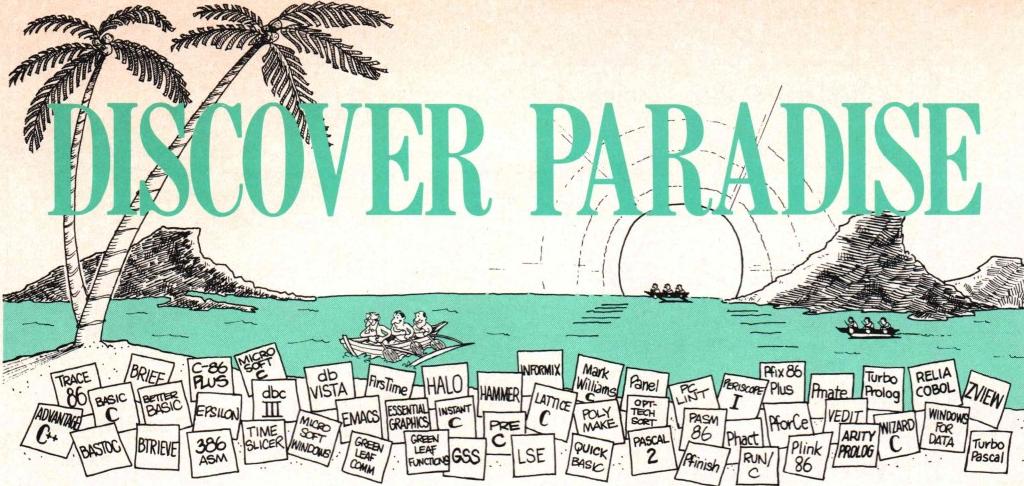
```

321             putchar('\r');
322         }
323     }
324
325 #endif
326 /*-----*/
327
328 #ifdef MAIN
329
330 typedef struct
331 {
332     char    str[16];
333     int     count;
334 }
335 STAB;
336
337 /*-----*/
338
339 getword( buf )
340 char   *buf;
341 {
342
343 #ifdef RANDOM /* ----- Generate 500 random words -----*/
344
345     static int      wordnum = 500;
346     int           num_letters, let;
347
348     if( --wordnum < 0 )
349         return 0;
350
351     while( (num_letters = rand() % 16) < 3 )
352         ;
353
354     while( --num_letters >= 0 )
355     {
356         let     = (rand() % 26) + 'a'; /* 26 letters in english */
357         *buf++ = (rand() % 10) ? let : toupper(let);
358     }
359
360 #else /* ----- Get words from standard input -----*/
361
362     int     c;
363
364     while( (c = getchar()) != EOF && !(isalnum(c) || c=='_') )
365         ;
366
367     if( c == EOF )
368         return 0;
369     else
370     {
371         *buf++ = c;
372         while( (c = getchar()) != EOF && !(isalnum(c) || c=='_') )
373             *buf++ = c;
374     }
375 #endif
376
377     *buf = '\0';
378     return 1;
379 }
380
381 /*-----*/
382
383 main( argc, argv )
384 {
385     char        word[80];
386     STAB       *sp;
387     HASH_TAB   *tabp;
388
389     tabp = maketab( 127 );
390
391     while( getword( word ) )
392     {
393         if( sp = (STAB *) findsym( tabp, word ) )
394         {
395             if( strcmp(sp->str,"123456789abcdef") != 0 )
396             {
397                 printf("NODE HAS BEEN ADULTERATED\n");
398                 exit( 1 );
399             }
400
401             sp->count++;
402         }
403     }

```

(continued on page 60)

LIST OURS



DISCOVER PARADISE

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The PC/MS-DOS software source that caters to your individual programming needs. Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Huge inventory, immediate shipment
- Latest versions
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee

We'll Match Any Nationally Advertised Price.

LIST OURS

| | LIST OURS |
|-------------------------------|-------------|
| C++ | |
| ADVANTAGE C++ | \$ 495 CALL |
| PFORCE++ | 395 CALL |
| C COMPILERS | |
| C-86 PLUS | 497 CALL |
| DATALIGHT - C | 60 49 |
| DATALIGHT - C DEVELOPER'S KIT | 99 79 |
| LATTICE C 3.2 | 500 289 |
| LATTICE C W/SOURCE | 900 545 |
| LET'S C | 75 59 |
| W/CSD DEBUGGER | 150 109 |
| MICROSOFT C 4.0 | 450 275 |
| MARK WILLIAMS C | 495 289 |
| SUPERSOFT C | 395 339 |
| WIZARD C | 450 369 |
| C INTERPRETERS | |
| C-TERP | 300 235 |
| INSTANT C | 500 379 |
| INTRODUCING C | 125 105 |
| RUN/C | 150 89 |
| RUN/C PROFESSIONAL 1.1 | 250 169 |
| ASSEMBLERS, LINKERS | |
| 386/AM/LINK | 495 445 |
| ADVANTAGE LINK | 395 349 |
| MACRO-86 | 150 98 |
| PASM-86 | 195 135 |
| PLINK 86 PLUS | 495 335 |
| QUELO 68000 X-ASM | 595 509 |

Polytron Specials

| | | |
|----------------------------------|----------|------|
| POLYBOOST | 80 | 65 |
| POLYTRON C BEAUTIFIER | 49 | 45 |
| POLYTRON C LIBRARY I | 99 | 75 |
| POLYTRON POWERCOM | 179 | 135 |
| POLYLIBRARIAN | 99 | 75 |
| POLYLIBRARIAN II | 149 | 115 |
| POLYMAKE | 99 | 75 |
| POLYWINDOWS PRODUCTS | CALL | CALL |
| POLYXREF | 219 | 175 |
| POLYXREF (One Language Only) | 129 | 105 |
| PVCS | 395 | 315 |
| GRAPHICS | | |
| ADVANTAGE GRAPHICS | 295 CALL | |
| ESSENTIAL GRAPHICS | 250 | 195 |
| GSS GRAPHICS DEVELOPMENT TOOLKIT | 495 | 379 |
| GSS KERNEL SYSTEM | 495 | 379 |
| GSS METAFILE INTERPRETER | 295 | 239 |
| GSS PLOTTING SYSTEM | 495 | 379 |
| HALO—ONE LANGUAGE | 300 | 209 |
| HALO—FIVE MICROSOFT LANGUAGES | 595 | 415 |
| METAWINDOWS | 185 | 115 |
| METALINDOWS PLUS | 235 | 189 |
| METAFONTS | 80 | 59 |
| METAFONTS PLUS | 235 | 189 |

Terms and Policies

- We honor MC, VISA, AMERICAN EXPRESS
- No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling \$3.00 per item, sent UPS ground. Rush service available, prevailing rates.
- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- Mention this ad when ordering — some items are specially priced.
- Prices and Policies subject to change without notice.
- Corporate and Dealer inquiries welcome.

1-800-445-7899 In NY: 1-800-642-6471

Circle no. 334 on reader service card.

PASCAL COMPILERS

| | | |
|------------------------|-----|-----|
| MICROSOFT PASCAL | 300 | 189 |
| PASCAL 2 | 350 | 329 |
| TURBO PASCAL | 100 | 69 |
| OTHER BORLAND PRODUCTS | | |

| | |
|------|------|
| CALL | CALL |
|------|------|

TOOLS FOR TURBO PASCAL

| | | |
|------------------------|-----|-----|
| ALICE | 95 | 68 |
| FIRSTIME | 75 | 59 |
| FLASH UP WINDOWS | 75 | 68 |
| HALO | 300 | 209 |
| TURBO HALO | 129 | 99 |
| SCREENPLAY | 100 | 89 |
| SCREEN SCULPTOR | 125 | 94 |
| T-DEBUG PLUS | 60 | 50 |
| TURBO EXTENDER | 85 | 65 |
| TURBO PASCAL ASYNC MGR | 100 | 84 |
| TURBO PROFESSIONAL | 70 | 49 |
| TURBO POWER TOOLS PLUS | 100 | 83 |
| TURBO WINDOWS | 80 | 65 |
| OTHER TURBO TOOLS | | |

| | |
|------|------|
| CALL | CALL |
|------|------|

NEW Products

ADVANTAGE GRAPHICS — Fast, powerful and extensive graphics library offering a full set of graphics primitives. No royalties, many languages! List \$295 Ours \$295

DATAWINDOWS — Greenleaf's latest offering includes integrated windows, transaction data entry, pop-up, pull-down, Lotus-style menu systems. And more! DataWindows is fast, writing directly to video memory. List \$225 w/Source List \$450 Ours \$159 Ours \$315

PASCAL 2 — Highly optimized Pascal compiler, with source level debugger, profiler. List \$350 Ours \$329

TIMESLICER — Multitasking, linkable library supporting concurrent tasks and real-time event processing with header files provided for C++, C and assembly. Library source available! List \$295 Ours \$265

VENTURA PUBLISHER (XEROX) — Desktop publishing software, lightning fast, loaded with features. Create professional-looking documentation at minimal cost! List \$895 Ours \$805

| BASIC | 199 | 139 |
|----------------------|------|-----|
| BETTERBASIC | 199 | 139 |
| SUMMIT ADD ONS | | |
| BETTER TOOLS | 95 | 89 |
| FINALLY | 99 | 89 |
| MICROSOFT QUICKBASIC | 99 | 75 |
| PROFESSIONAL BASIC | 99 | 75 |
| 8087 MATH SUPPORT | 50 | 45 |
| PANEL BASIC | 145 | 115 |
| TRUE BASIC | 150 | 105 |
| ADD ONS | | |
| CALL | CALL | |

OTHER PRODUCTS AVAILABLE TO THE BASIC PROGRAMMER INCLUDE MULTITHALO, BTREIVE, GSS GRAPHICS, SCREEN SCULPTOR, STRUBAS, 87 BASIC.

COBOL COMPILERS/UTILITIES

| | | |
|-----------------------|------|------|
| MICROSOFT COBOL | 700 | 445 |
| MICROSOFT COBOL TOOLS | 350 | 205 |
| MICROSOFT SORT | 195 | 139 |
| MICRO/SPF | 175 | CALL |
| OPT-TECH SORT | 149 | 115 |
| REALIA COBOL | 995 | 785 |
| SCREENPLAY | 175 | 155 |
| RM/COBOL | 950 | 639 |
| RM/COBOL 8X | 1250 | 895 |
| VISUAL COBOL (MBP) | 1150 | 1015 |

FORTRAN COMPILERS/UTILITIES

| | | |
|--|-----|------|
| LAHEY FORTRAN | 477 | CALL |
| MICROSOFT FORTRAN | 350 | 209 |
| RM/FORTRAN | 595 | 389 |
| ACS TIMES SERIES | 495 | 419 |
| 87 SFL | 250 | 225 |
| FOR-WINDS | 90 | 78 |
| FORLIB-PLUS | 70 | 54 |
| GRAFMATICS OR PLOTMATICS | 135 | 119 |
| GRAFMATICS AND PLOTMATICS | 240 | 219 |
| FORTRAN SCIENTIFIC SUBROUTINES | 295 | 249 |
| POLYFORTRAN TOOLS I | 179 | 143 |
| STRINGS AND THINGS | 70 | 54 |
| ALSO AVAILABLE TO THE FORTRAN PROGRAMMER: PANEL, MULTITHALO, BTREIVE, ESSENTIAL GRAPHICS, FLASH UP WINDOWS, GSS GRAPHICS, OPT-TECH SORT. | | |

PROLOG

| | | |
|--|------|------|
| ARITY PROLOG (STANDARD) | 95 | 79 |
| ADDIT. ARITY PRODUCTS | CALL | CALL |
| CHALCEDONY PROLOG | 100 | 89 |
| TURBO PROLOG | 100 | 79 |
| LISP, OTHER AI, CALL FOR INFORMATION, PRICING, AVAILABILITY. | | |

February's Bundle of the Month

BTREIVE, XTRIEVE, REPORT GENERATOR (formerly RTRIEVE) LISTS TOGETHER \$635 OURS \$495!!!

| EDITORS | 195 | CALL |
|--------------|-----|------|
| BRIEF | 75 | 59 |
| CVUE | 250 | 195 |
| W/SOURCE | | |
| EDIX | 195 | 155 |
| EMACS | 295 | 265 |
| EPSILON | 195 | 159 |
| FIRSTIME (C) | 295 | 229 |
| KEDIT | 125 | 105 |
| LSE | 125 | 95 |
| PMATE | 195 | 125 |
| PC/VII | 149 | 129 |
| SPF/PC | 195 | 149 |
| VEDIT | 150 | 109 |
| VEDIT PLUS | 185 | 139 |

| | | |
|-----------------------------|-----|-----|
| DAN BRICKLIN'S DEMO PROGRAM | 75 | 59 |
| FASTBACK | 175 | 149 |
| INTERACTIVE EASYFLOW | 150 | 129 |
| PDISK | 195 | 129 |
| SOURCE PRINT | 139 | 115 |
| VENTURA PUBLISHER (XEROX) | 895 | 805 |

Programmer's Paradise

487 E. Main Street, Mt. Kisco, NY 10549
914-332-4548

Programmer's Paradise



**MAKE YOUR PC
SEEM LIKE AN AT!**

**MAKE YOUR AT
SEEM LIKE A
DREAM MACHINE!**



The Integrated Console Utility™
**FAST, POWERFUL
ANSI.SYS REPLACEMENT**

For the IBM-PC, AT, and clones

New Version 2.00 is MUCH FASTER!
Now blink free scrolling on CGA!

Now uses EMS/EEMS for Scroll Recall
New Menu Program for Changing Options

**GET A BOX FULL OF UTILITIES!
MAKE LIFE EASIER FOR ONLY \$75!**

- Speed up your screenwriting 2-6x
- Extend your ANSI.SYS to full VT100
- Add many more escape sequences
- Scroll lines back onto screen
- Save scrolled lines into a file
- Add zip to your cursor keys
- Free your eyes from scroll blinking
- Easy installation
- Get a 43 line screen w/EGLA
- Get a 50 line screen w/CGA
- No more annoying typeahead beep
- Prevent screen phosphor burnin
- Control many programs' use of color
- Generate breakpts from keyboard
- Shorten that annoying bell
- Over 50 other useful options

"The psychological difference is
astonishing"

—Lotus June 85 pg 8.

"So many handy functions rolled into
one unobtrusive package"
—PC-World Feb 86 pg 282.

"The support provided by the
publishers is extraordinary"
—Capital PC Monitor May 86 pg 25.

"...the best choice for improving your
console..."
—Capital PC Monitor June 86 pg 26.

460p Manual (w/slip case) & disks \$75.

**Satisfaction Guaranteed!
Order Yours Today!**

HERSEY MICRO CONSULTING
Box 8276, Ann Arbor, MI 48107
(313) 994-3259 VISA/MC/Amex

DEALER INQUIRIES INVITED



Circle no. 280 on reader service card.

C CHEST

Listing Four (Listing continued, text begins on page 90.)

```
404 {  
405     sp = (STAB *) addsym( tabp, word, sizeof(STAB) );  
406     strcpy( sp->str, "123456789abcdef" );  
407     sp->count = 1;  
408 }  
409 }  
410 pstats( tabp ); /* Print statistics */  
412  
413 #ifdef DEBUG  
414     dptab( tabp );  
415 #endif  
416 }  
417  
418 #endif
```

End Listing Four

Listing Five

```
1 /* ITOASCII.C  
2 *  
3 *      Routines to convert integers to ascii strings.  
4 *      itoroman converts to a roman numeral.  
5 *      itoalpha converts to a string of letters such as are used  
6 *          in an outline.  
7 *      itoascii does all of the above + arabic format.  
8 *  
9 *  
10 *      None of these routines check for string overflow. The  
11 *      largest number of characters generated by the routines  
12 *          are:  
13 *          format    routine    longest #        #digits  
14 *          alpha     itoalpha    AVLG            4  
15 *          roman    itoroman   MMMMDCCCLXXXVIII   16  
16 *          arabic   ----       32767           5  
17 *          total:   itoascii  
18 *  
19 *          The exception to the above is "english" format where the  
20 *          longest string is:  
21 *          "minus twenty seven thousand seven hundred seventy-seven"  
22 *          56 characters including the terminating null.  
23 *          Allowing an extra space for a potential leading - sign  
24 *          and another for a terminating blank we get a worst case  
25 *          of 18 characters.  
26 */  
27  
28 #define SSIZE 16 /* Maximum size of expanded number */  
29  
30 extern char *cpy (char*, char* );  
31 extern void strcpy (char*, char* );  
32  
33 /*-----*/  
34  
35 itoeng( dest, uppercase, n )  
36 int n, uppercase;  
37 char *dest;  
38 {  
39     /* Convert number to english string "one", "two", etc. If  
40     * uppercase then the first letter is capitalized. The  
41     * number of characters in the string is returned.  
42     * Longest possible string is:  
43     *          "minus twenty seven thousand seven hundred seventy seven"  
44     *          (56 characters including the null).  
45     */  
46  
47     static char *onestab[] =  
48     {  
49         "zero",    "one",      "two",      "three",    "four",  
50         "five",    "six",      "seven",    "eight",    "nine",  
51         "ten",     "eleven",   "twelve",   "thirteen", "fourteen",  
52         "fifteen", "sixteen",  "seventeen", "eighteen", "nineteen"  
53     };  
54  
55     static char *tenstab[] =  
56     {  
57         "",        "ten",     "twenty",   "thirty",   "forty",  
58         "fifty",   "sixty",   "seventy",  "eighty",   "ninety"  
59     };  
60  
61     char *first;  
62  
63     first = dest;  
64  
65     if( n == 0 )  
66     {
```

EditingTools

version 2.0

A Superb
Text Editor
with an Intelligent
DOS Shell

```
67         if( uppercase )
68             cpy( dest, uppercase ? "Zero" : "zero" );
69
70     return( 5 );
71 }
72
73 if( n < 0 )
74 {
75     dest = cpy( dest, "minus " );
76     n = -n;
77 }
78
79
80 if( 20000 > n >= 10000 )
81 {
82     dest = cpy( dest, onestab[n / 1000] );
83     dest = cpy( dest, (n % 1000) ? " thousand, " : " thousand");
84 }
85 else if( n >= 1000 )
86 {
87     if( n >= 20000 )
88     {
89         dest = cpy( dest, (n >= 30000) ? "thirty" : "twenty");
90         if( n % 1000 )
91             dest = cpy( dest, (n >= 1000) ? "-" : " " );
92     }
93
94     if( n >= 1000 )
95     {
96         dest = cpy( dest, onestab[n / 1000] );
97         dest = cpy( dest, " " );
98     }
99
100    dest = cpy( dest, (n % 1000) ? "thousand, " : "thousand");
101
102
103 if( n >= 100 )
104 {
105     dest = cpy( dest, onestab[ n/100 ] );
106     dest = cpy( dest, (n % 100) ? " hundred " : " hundred");
107 }
108
109 if( n >= 20 )
110 {
111     dest = cpy( dest, tenstab[ n / 10 ] );
112     if( n % 10 )
113         dest = cpy(dest, "-");
114 }
115
116 if( n )
117     dest = cpy( dest, onestab[n] );
118
119 if( uppercase )
120     *first = toupper( *first );
121
122 return( (dest - first) +1 );
123 }
124 }
125 */
126 /*-----*/
127
128 itoroman( dest, uppercase, n )
129 int      n, uppercase;
130 char    *dest;
131 {
132     /*
133     * Convert integer to a Roman numeral.
134     * Return number of characters put into dest.
135     *
136     * Bugs: 1) Numbers larger than 4999 are not printed. This
137     *        is because characters are required which don't
138     *        exist in the ASCII character set (ie. letters
139     *        with lines over them.
140     *
141     *        2) The number 0 is represented as a '0' which didn't
142     *        exist in Roman numerals. Similarly, negative
143     *        numbers have a preceeding "-" sign.
144     */
145
146 register char  *cp, **rp, *start;
147 register int   i ;
148
149 static char   *rnums[] =
150 {
```

(continued on next page)

EditingTools Editor

Fast, powerful, simple to use. Edit multiple files, limited only by available memory. Move text between files. Deleted lines are recoverable from the trash file.

EditingTools DOS Shell

An intelligent interface between DOS and the Editor. Load multiple directories as menus in easy to read table format - names of files with the same extension are sorted into a column, labeled by their common extension. Select a file to edit, or a program to execute without worrying about paths.

Shell Commands

Execute DOS command, Change color/ drive, ChDir, Copy, Delete, Edit, Execute, Rename, Load/ erase directory, Previous/ next directory, List commands, Store/ recall command, Return to editor, and much more.

Editor Commands

Auto insert/ indent, Indent block, Previous/ next file, Edit trash file, Verify key, Restore line, Find/ replace, Goto line, Repeat/ reverse goto, Tab right/ left, List commands, Blank screen, Return to shell, and much more.

Source Code

Over 10,000 lines in Turbo Pascal. Since Turbo Pascal is not an optimizing compiler, almost all procedures are optimized manually in assembly inline code.

System Requirements

EditingTools is only 38K in size, and runs efficiently on IBM PC, XT, AT and true compatibles. No installation program. All editor command keys can be changed effortlessly during editing. It is not copy protected.

Incredible Value

Add EditingTools to your programming toolbox for only \$30. Add \$25 for well-written source code. 60 day money back guarantee. Please add \$2 for s/h. To order, send check or money order to:

Dr. Jiann Jou
P.O.Box 460969
Garland, TX 75046
(214) 495-8862

Circle no. 355 on reader service card.

DDJ 1986 Back Issues

March 1986 #113 Volume XI, Issue 3

Parallel Processing—Concurrency and Turbo Pascal—What Makes DOS Fast—Minimizing Arbitrary Functions—MC68000 vs. NS32000.

April 1986 #114 Volume XI, Issue 4

Special AI Issue—Programming in LISP and Prolog—An Expert at Life—Perils of Protected Mode—I/O Redirection for the Shell.

May 1986 #115 Volume XI, Issue 5

Software Design from the Outside In—Dan Bricklin's DEMO Program—Crytographer's Toolbox—EGA Graphics & Fast PC Graphics—How to Write Memory Resident Code.

June 1986 #116 Volume XI, Issue 6

Telecommunications Without Errors—General-Purpose Sorting—Structured Programming.

July 1986 #117 Volume XI, Issue 7

Special FORTH Issue—Forth Standards Proposal—Forth in a Bottle—Forth & Expanded Memory—Forth Structured Programming.

Aug. 1986 #118 Volume XI, Issue 8

Special C Issue—Benchmarking C Compilers—The Joy of Conciseness—Nearly Perfect Trees—Generics in Ada—Real-World Data Types.

Sept. 1986 #119 Volume XI, Issue 9

Smooth Algorithms—MS-DOS Directory Traversal—Turbo Boards Review—Radix Sort—Does Turbo Prolog Measure Up—Crawling Memory Test.

Oct. 1986 #120 Volume XI, Issue 10

80386 Programming—MS-DOS File Browsing—Converting to the 320xx—Modula-2 Compiler Review—Factoring in Forth.

Nov. 1986 #121 Volume XI, Issue 11

Graphics Routines—The New Graphics Chips—Programming Tips in C, Modula-2, Pascal, and Ada—68k Graphics.

Dec. 1986 #122 Volume XI, Issue 12

Multitasking—32000 Assembler—Comparing String Comparisons—Turbo Pascal Procedural Parameters.

Other issues are also available. Please inquire.

TO ORDER: Return this coupon with your payment to:
M&T Books, 501 Glaveston Dr. Redwood City, CA 94063.
Or, call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m.
In CA call 800-356-2002

Please send the issues circled:

113 114 115 116 117 118
119 120 121 122 123 124

Price: 1 issue-\$5.00. 2-5 issues-\$4.50 each. 6 or more-\$4.00 each. (There is a \$10.00 minimum for charge orders.)

Subtotal _____

CA residents add sales tax _____ % _____

Outside U.S., add \$.50 per issue _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.

Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

C CHEST

Listing Five (Listing continued, text begins on page 90.)

```

151      "",      "C",      "CC",      "CCC",      "CD",
152      "D",      "DC",      "DCC",      "DCCC",      "CM",
153
154      "",      "X",      "XX",      "XXX",      "XL",
155      "L",      "LX",      "LXX",      "LXXX",      "XC",
156
157      "",      "I",      "II",      "III",      "IV",
158      "V",      "VI",      "VII",      "VIII",      "IX"
159  };
160
161  start = dest;
162
163  if( n <= -5000 || n >= 5000 ) /* Number can't be represented */
164  {
165      strcpy( dest, "*****" );
166      return 0;
167  }
168
169  if( n == 0 ) /* Deal with a zero */
170      *dest++ = '0';
171
172  else if( n < 0 ) /* Print preceding - if necessary */
173  {
174      *dest++ = '-';
175      n = -n;
176  }
177
178  while( n >= 1000 ) /* Take care of leading thousands */
179  {
180      *dest++ = (uppercase) ? 'M' : 'm' ;
181      n -= 1000;
182  }
183
184  rp = rnums;
185  for( i = 10*10 ; n>0 && i>=1 ; i/=10 )
186  {
187      cp = *(rp + (n/i)); /* Find the appropriate str.*/
188      n %= i;
189      rp += 10;
190
191      for( *cp ; cp++)
192          *dest++ = (uppercase) ? *cp : *cp + ('a'-'A') ;
193  }
194
195  *dest = '\0' ;
196  return(dest - start);
197 }
198
199 /*****
200
201 202 itoalpha(dest, uppercase, n)
202 int      n, uppercase ;
203 char    *dest;
204
205 /*
206     Convert integer to an ASCII string as follows:
207
208     *
209     *          0, a, b, c, ... y, z, aa, ab, ac, ad ...
210
211     *
212     *          Return number of characters in expanded string.
213
214     *
215     *          This routine is very similar to itoa(). The problem here
216     *          is that the above sequence is not a base 26 number.
217     *          That is, there is no equivalent to a zero.
218     *          (z + 1 = aa (if a=0, b=1 ... then z + 1 would = ba )
219     *          or to look at it another way, if a=0, b=1, etc. then
220     *          aa should equal 0. It doesn't.
221
222     */
223
224     char      scratch[SSIZE+1], *p, *start ;
225
226     start = dest;
227
228     if( n < 0 )
229     {
230         *dest++ = '-';
231         n = -n;
232     }
233
234     if( n == 0 )
235         *dest++ = '0';
236     else
237     {

```

```

234         --n ;
235         p = scratch;
236
237         do{
238             *p++ = (n % 26) + (uppercase ? 'A' : 'a') ;
239
240         } while( (n = (n/26)-1) >= 0 );
241
242         while( --p >= scratch )           /* Copy scratch space to */
243             *dest++ = *p ;                /* destination string. */
244     }
245
246     *dest = '\0' ;
247     return(dest - start);
248 }
249
250 /*-----*/
251
252 itoascii( str, fmt, n )
253 char    *str;
254 {
255     /* convert integer "n" to an ascii string according to
256     * "fmt" and put it in "str";
257     *
258     *      fmt == 'i'      lower case roman numerals
259     *      fmt == 'I'      upper case roman numerals
260     *      fmt == 'a'      lower case alphabetic
261     *      fmt == 'A'      upper case alphabetic
262     *      fmt == 'e'      spelled out in lower case.
263     *      fmt == 'E'      spelled out w/ 1st char capitalized.
264     *      fmt == '1'      arabic, field zero padded to 1 char
265     *      fmt == '2'      arabic, field zero padded to 2 char
266     *
267     *      Return width of string in str.
268 */
269
270     register int    rval    = 0 ;
271
272     register char   *format = "%0ld" ;
273     register char   *mfmt   = "-%0ld" ;
274
275     if( '0' <= fmt && fmt <= '9' )
276     {
277         if( n < 0 )
278         {
279             /* This kludge makes up for a bug in sprintf.
280             * sprintf( str, "%04d", -2 ) loads str with:
281             * "000-2"
282             */
283
284             mfmt[3] = fmt;
285             n = -n;
286             rval = sprintf( str, mfmt, n );
287         }
288         else
289         {
290             format[2] = fmt;
291             rval = sprintf( str, format, n );
292         }
293     }
294     else if( fmt == 'i' || fmt == 'I' )
295     {
296         rval = itoroman( str, fmt == 'I', n );
297     }
298     else if( fmt == 'a' || fmt == 'A' )
299     {
300         rval = itoalpha( str, fmt == 'A', n );
301     }
302     else if( fmt == 'e' || fmt == 'E' )
303     {
304         rval = itoeng( str, fmt == 'E', n );
305     }
306
307     return( rval );
308 }
309
310 #ifdef DEBUG
311
312 main()
313 {
314     int      n;
315     char    str[80];
316
317     while( 1 )
318     {

```

(continued on next page)

STAT Toolbox for Turbo Pascal

Bring convenience, power and versatility to your statistics programs!

Two statistical packages in one!

A library disk and reference manual

Use these powerful statistical routines to build your applications. Routines include: • statistical distribution functions • random-number generation • basic descriptive statistics • parametric and non-parametric statistical testing • bivariate linear regression, multiple and polynomial regression.

A demonstration disk and manual

This package incorporates the library of routines into a fully functioning statistical program. Two data management programs are included to facilitate the storage and maintenance of data.

Full source code is included. (For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC-DOS 2.0 or later are required.)

STAT Toolbox

Item #050 \$69.95



TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the **STAT Toolbox** for only \$69.95

Subtotal _____

CA residents add sales tax _____ % _____

Add \$2.25 per item for shipping _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

SK:

The System Kernel of the Tele Operating System Toolkit

Tele is a multitasking operating system written in C and assembly language for IBM PC compatibles. **SK: The System Kernel** of the Tele Operating System is now available, including the most crucial part of Tele—the task scheduling algorithm first published in *Dr. Dobb's* December 1986 issue. The System Kernel is required by the Console system available in March, and by the soon-to-be-released File and Index systems of the Tele Operating System Toolkit. When all components of the Toolkit are integrated they form an independent operating system for any 8086-based machine. Tele has also been designed for compatibility with MS-DOS, Unix and the MOSI standard.

SK: The System Kernel contains an initialization module, general purpose utility functions, and a real time task management system. **SK: The System Kernel** provides MS-DOS applications with multitasking capabilities. Only \$49.95!

SK: the System Kernel includes all C and assembler source code, and precompiled libraries! MS-DOS disk format



TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me **SK: The System Kernel** for only \$49.95

CA residents add sales tax _____ % _____

In U.S. add \$2.25 per item for shipping
& handling _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

C CHEST

Listing Five (Listing continued, text begins on page 90.)

```

319     printf("Enter a decimal number: ");
320     scanf("%d", &n);
321     itoascii( str, '9', n );      printf("%s\n", str );
322     itoascii( str, '0', n );      printf("%s\n", str );
323     itoascii( str, 'a', n );      printf("%s\n", str );
324     itoascii( str, 'A', n );      printf("%s\n", str );
325     itoascii( str, 'i', n );      printf("%s\n", str );
326     itoascii( str, 'I', n );      printf("%s\n", str );
327     itoascii( str, 'e', n );      printf("%s\n", str );
328     itoascii( str, 'E', n );      printf("%s\n", str );
329   }
330 }
331
332 #endif

```

End Listing Five

Listing Six

```

1 #include <stdio.h>
2 #include <ctype.h>
3 #include <stdarg.h>
4
5 /* PARSE.C           Expression parser for infix desk calculator
6 *
7 * Copyright (c) 1986, Allen I. Holub. All rights reserved.
8 *
9 * General purpose expression analyzer. Can evaluate any expression
10 * consisting of number and the following operators (listed according
11 * to precedence level):
12 *
13 *      () - ! 'str' 'str'
14 *      * / %
15 *      +
16 *      < <= > >= == !=
17 *      && ||
18 *
19 * In a string (ie, in strcmp()) * and ? work as in DOS.
20 * All operators associate left to right unless () are present.
21 * The top - is a unary minus. exists() evaluates to true if the
22 * filename exists. strcmp() evaluates to true if the strings match.
23 * isfile() evaluates to true if the file exists and is a file (as
24 * compared to a directory). All whitespace is ignored and " counts
25 * as whitespace.
26 *
27 * <expr> ::= <term> <expr>
28 * <expr> ::= && <term> <expr>
29 *          ::= || <term> <expr>
30 *          ::= epsilon
31 *
32 * <term> ::= <fact> <term1>
33 * <term1> ::= < <fact> <term1>
34 *          ::= <= <fact> <term1>
35 *          ::= > <fact> <term1>
36 *          ::= >= <fact> <term1>
37 *          ::= == <fact> <term1>
38 *          ::= != <fact> <term1>
39 *          ::= epsilon
40 *
41 * <fact> ::= <part> <fact1>
42 * <fact1> ::= + <part> <fact1>
43 *          ::= - <part> <fact1>
44 *          ::= epsilon
45 *
46 * <part> ::= <const> <part1>
47 * <part1> ::= * <const> <part1>
48 *          ::= / <const> <part1>
49 *          ::= epsilon
50 *
51 * <const> ::= ( <expr> )
52 *          ::= - ( <expr> )
53 *          ::= - <const>
54 *          ::= ! <const>
55 *          ::= 's1's2'          (* like strcmp(s1,s2) *)
56 *          ::= NUMBER
57 *
58 * Note that several productions are combined in the following
59 * subroutines. For example <expr> and <expr1> are combined into
60 * a single subroutine. Similarly, all right recursion has been
61 * replaced with "for" loops. External subroutines are:
62 *
63 *      VTYPE parse( expr_p ) char **expr_p ;
64 *
65 * which parses the expression in the string pointed to by *expr_p
66 * and returns the parsed value. VTYPE (defined below) is currently
67 * a double but can be changed by modifying the typedef and

```

```

68 *      recompiling. *expr_p is advanced past the expression. Parsing will
69 *      stop when the first character that can't be part of an expression
70 *      is encountered. This includes numbers as in: "3+5 6"
71 *      where parse will return 8 and *expr_p will be pointing at the '6'.
72 *
73 *          int      startexpr( str )      char *str;
74 *
75 *      returns true if if the first token in str is in FIRST(expr).
76 *
77 */
78 */
79
80 extern double  atof( char * );
81 extern long    atol( char * );
82
83 typedef double VTYPE;           /* Static global varialbes */
84
85 static  char   *Str;
86 static  char   *Start_str;
87
88 int     find   (int      );      /* local static subroutines */
89 int     match  (char * );
90 void    error  (char *, ... );
91 VTYPE   expr   ();
92 VTYPE   term   ();
93 VTYPE   fact   ();
94 VTYPE   part   ();
95 VTYPE   const  ();
96 VTYPE   constant();
97
98 #define      advance(amt)  Str+=(amt)
99
100 /*
101 int     startexpr( str )
102 char   *str;
103 {
104     /* Returns true if the first token in str is in FIRST(expr).
105     */
106
107     register int c;
108
109     c = *str;
110
111     return( isdigit(c) || c=='(' || c=='-' || c=='.'
112         || c=='!' || c=='\'' );
113 }
114
115 */
116
117 static int    find( c )           /* Advance Str to c or to EOS */
118 {
119     while( *Str && *Str != c )
120         Str++;
121
122     if( !*Str )
123     {
124         error("missing %c in expression", c );
125         return 0;
126     }
127     return 1;
128 }
129
130 */
131
132 static int    match( token )
133 char   *token;
134 {
135     register char  *p;
136
137     while( isspace(*Str) || *Str == ' ' )
138         Str++;
139
140     for(p = Str; *token && *token == *p ; p++, token++)
141         ;
142
143     return( *token == '\0' );
144 }
145
146 */
147
148 static void error( fmt )        /* has a variable number of args */
149 char   *fmt;
150 {
151     register char  *p;
152     va_list  args;

```

(continued on next page)

TURBO Advantage:

Source Code Libraries for Turbo Pascal

This library of more than 220 routines, complete with source code, sample programs and documentation will save you hours developing and optimizing your programs!

Routines are organized and documented under the following categories: bit manipulation, file management, MS-DOS support, sorting, string operations, arithmetic calculations, data compression, differential equations, Fourier analysis and synthesis, matrices and vectors, statistics, and much more! All source code is included.

A detailed manual includes a description of the routine, an explanation of the methods used, the calling sequence, and a simple example. For MS/PC-DOS systems.

TURBO Advantage: Source Code Libraries for Turbo Pascal is also available with **TURBO Advantage Complex: Complex Number Routines for Turbo Pascal** and **TURBO Advantage Display: Form Generator for Turbo Pascal**.

Turbo Advantage Item #070 \$49.95



M&T BOOKS

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the **Turbo Advantage: Source Code Libraries for Turbo Pascal** for only \$49.95

Subtotal _____

CA residents add sales tax _____ % _____

Add \$2.25 per item for shipping _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

TURBO Advantage Display:

**Form Generator
for Turbo Pascal**

Now, even if you have little programming knowledge, you can design and process forms to fit your needs!

TURBO Display includes a menu driven form processor, 30 Turbo Pascal procedures and functions to facilitate linking created forms to your program, and full source code and documentation. For MS-DOS systems.

Some of the *TURBO Advantage: Source Code Libraries for Turbo Pascal* routines are necessary to compile *TURBO Display*. You save \$20 when you order *TURBO Advantage: Source Code Libraries for Turbo Pascal* together with *TURBO Display: Form Generator for Turbo Pascal!* Receive both for only \$99.95!

TURBO Display: Form Generator for Turbo Pascal is also available individually for \$69.95.

**TURBO Advantage/
Display Package** Item #070B \$99.95
TURBO Display Item #072 \$69.95



TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the *Turbo Advantage/
Display Package* for only \$99.95

Send me *Turbo Display: Form
Generator for Turbo Pascal* for \$69.95

Subtotal _____

CA residents add sales tax _____ % _____

Add \$2.25 per item for shipping _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

C CHEST

Listing Six (Listing continued, text begins on page 90.)

```

153     va_start( args, fmt );
154
155     printf("%s\n", Start_str );
156
157     for( p = Start_str; p < Str; p++ )
158         printf("_");
159
160     printf( "^\n" );
161     vprintf( fmt, args );
162     printf( "\n" );
163
164 }
165
166
167 /*-----*/
168
169 static VTYPE expr()
170 {
171     VTYPE left;
172
173     left = term();
174
175     for(;;)
176     {
177         if      ( match("&&") ) { advance(2); left = term() && left; }
178         else if( match("||") ) { advance(2); left = term() || left; }
179         else break;
180     }
181
182     return left;
183 }
184
185 /*-----*/
186
187 static VTYPE term()
188 {
189     VTYPE left;
190
191     left = fact();
192
193     for(;;)
194     {
195         if      ( match("<=") ) { advance(2); left = left <= fact(); }
196         else if( match("<") ) { advance(1); left = left < fact(); }
197         else if( match(">=") ) { advance(2); left = left >= fact(); }
198         else if( match(">") ) { advance(1); left = left > fact(); }
199         else if( match("==") ) { advance(2); left = left == fact(); }
200         else if( match("!=") ) { advance(2); left = left != fact(); }
201         else break;
202     }
203
204     return left;
205 }
206
207 /*-----*/
208
209 static VTYPE fact()
210 {
211     VTYPE left;
212
213     left = part();
214
215     for(;;)
216     {
217         if      ( match("+") ) { advance(1); left += part(); }
218         else if( match("-") ) { advance(1); left -= part(); }
219         else break;
220     }
221
222     return left;
223 }
224
225
226 /*-----*/
227
228 static VTYPE part()
229 {
230     VTYPE left;
231     static tmp;
232
233     left = const();
234
235     for(;;)

```

```

236     {
237         if( match("*") )
238         {
239             advance(1);
240             left *= part();
241         }
242         else if( match("%") )
243         {
244             advance(1);
245             left = (long)left % (long)part();
246         }
247         else if( match("/") )
248         {
249             advance(1);
250             if( tmp = part() )
251                 left /= tmp;
252             else
253                 error( "Divide by 0\n" );
254         }
255         else break;
256     }
257
258     return left;
259 }
260
261 /*-----*/
262
263 static VTYPE const()
264 {
265     register VTYPE rval = 0;
266     int sign = 1, logical_not = 0;
267     static char *s1, *s2;
268
269     if( match("-") ) { advance(1); sign = -1; }
270     if( match("!) ) { advance(1); logical_not = 1; }
271
272     if( match("(") )
273     {
274         advance(1);
275         rval = expr();
276
277         if( match(")") )
278             advance(1);
279         else
280             error("Mis-matched parenthesis\n");
281     }
282     else if( match("\\" ) )
283     {
284         s1 = ++Str;
285
286         if( !find( '\\" ) ) goto abort;
287         *Str++ = 0;
288         s2 = Str;
289
290         if( !find( '\\" ) ) goto abort;
291         *Str++ = 0;
292
293         rval = strcmp( s1, s2 );
294
295         s2 [-1] = '\\" ;
296         Str[-1] = '\\" ;
297     }
298     else
299     {
300         rval = ( sizeof(VTYPE) == sizeof(double) )
301             ? atof( Str )
302             : (VTYPE) atol( Str )
303             ;
304
305         while( isdigit(*Str) || *Str == '.' )
306             Str++;
307     }
308
309 abort:
310     return( logical_not ? !rval : rval * sign );
311 }
312
313 /*-----*/
314
315 VTYPE parse( expr_p )
316 char **expr_p;
317 {
318     /* Return the value of "expression" or 0 if any errors were
319      * found in the string. "*Err" is set to the number of errors.

```

(continued on next page)

TURBO Advantage Complex:

Complex Number Routines for Turbo Pascal

Working with complex numbers is easy with the Turbo Pascal procedures and routines provided in TURBO Advantage Complex!

TURBO Complex provides procedures for performing all the arithmetic operations and necessary real functions with complex numbers. Each procedure is based on predefined constants and types. By using these declarations the size of arrays are easily adapted. Each type declaration is a record with both a real and imaginary part. Use these procedures to build more sophisticated functions in your own programs.

TURBO Complex also demonstrates the usage of these procedures in routines for vector and matrix calculation with complex numbers and variables; simultaneous Fourier transforms; calculations of convolution and correlation functions; low-pass, high-pass, band-pass and band-rejection digital filters; and solving linear boundary-value problems.

Source code and documentation is included. For MS-DOS systems. Some of the *TURBO Complex* routines are most effectively used with routines contained in *TURBO Advantage*. Receive both *TURBO Advantage* and *TURBO Complex*, together for only \$115! You save \$25!

TURBO Complex: Complex Number Routines for Turbo Pascal is also available individually for \$89.95.

| | |
|-------------------------|--------------------------|
| TURBO Advantage/ | |
| Complex Package | Item #070A \$115 |
| TURBO Complex | Item #071 \$89.95 |

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the **Turbo Advantage/**
Complex Package for only \$115

Send me **Turbo Complex: Complex Number Routines** for \$89.95

Subtotal _____

CA residents add sales tax _____ %

Add \$2.25 per item for shipping _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

The Turbo Pascal Toolbook

Edited by
Namir Clement Shammas

Make your programming easier and more powerful with the Turbo Pascal Toolbook!

You'll find:

- an extensive library of low-level routines
- external sorting and searching tools, presenting a new database routine that combines the best features of the B-tree, B+ and B++ trees
- window management, to help you create, sort and overlay windows
- artificial intelligence techniques
- mathematical expression parsers, offering two routines that convert mathematical expressions into RPN tokens
- a smart statistical regression model that searches for the best regression model to represent a given set of data.

All routine libraries and sample programs are on disk for MS-DOS systems, and over 800K of Turbo Pascal source code is included!

Turbo Pascal
Toolbook Item #080 \$25.95
Turbo Pascal Toolbook
with disk Item #081 \$45.95

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the Turbo Pascal
Toolbook for only \$25.95

Send me the Toolbook, along with the
disk for only \$45.95

Subtotal _____

CA residents add sales tax _____ % _____

Add \$2.25 per item for shipping _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

C CHEST

Listing Six (Listing continued, text begins on page 90.)

```
320     * "Parse" is the "access routine" for expr(). By using it you
321     * need not know about any of the global variables used by expr().
322     */
323
324     VTYPE rval;
325
326     Start_str = Str = *expr_p ;
327
328     if( !Str || !*Str )
329         return 0;
330
331     rval = expr();
332
333     *expr_p = Str;
334     return rval;
335 }
336
337 /*****
338 #ifdef MAIN
339
340 main( argc, argv )
341 char    **argv;
342 {
343     /*      Desk calculator program to test parse(). If no cmd line
344     *      arguments are present, works in interactive mode and
345     *      exits with a 0 status. If command line arguments are
346     *      present, the expressions (one per argv entry) are
347     *      evaluated and the result printed. The exit status is
348     *      the result of the last expression evaluated, truncated
349     *      to unsigned char (0-255). If -s is specified, nothing
350     *      is printed but the same exit status is returned, you
351     *      can then test $status in a shell script to get the
352     *      status. For example, the following script prints "50":
353
354     *          expr -s "10 * 5"
355     *          echo $status
356
357
358     char    buf[128], *p ;
359     int    silent = 0;
360     VTYPE   rval;
361
362     if( argc == 1 )
363     {
364         printf("Enter expression or blank line to exit\n");
365
366         for( ; gets(buf); printf(">") )
367         {
368             if( !buf )
369                 break;
370
371             if( !startexpr(buf) )
372                 printf( "%s not an expression\n", buf );
373
374             p = buf;
375             printf( "%g\n", parse( &p ) );
376 #ifdef DEBUG
377             printf("tail = <%s>\n", p );
378 #endif
379         }
380     }
381     else
382     {
383         if( argv[1][0] == '-' && argv[1][1] == 's' )
384         {
385             silent = 1;
386             argv++;
387             argv--;
388         }
389
390         for( p = *++argv; --argc > 0 ; p = *++argv )
391         {
392             rval = parse( &p );
393             if( !silent )
394                 printf( "%s = %g\n", *argv, rval );
395         }
396     }
397
398     exit( (unsigned char) rval );
399 }
400
401 #endif
```

End Listing Six

ON COMMAND:

Writing a Unix-Like Shell for MS-DOS

by Allen Holub

Listing Seven

```

1 #include <ctype.h>
2
3 char *skipspace( p, esc )
4 register char *p ;
5 register int esc ;
6 {
7     /*
8      * Skip all ' ' characters. \<space>, where \ is the
9      * "esc" character, doesn't count as a space.
10     */
11
12     while( *p == ' ' )
13     {
14         if ( *p != esc )
15             p++ ;
16
17         else if ( *++p ) /* skip escaped characters */
18             p++ ;
19     }
20
21     return(p) ;
22 }

```

End Listing Seven

Listing Eight

```

1 char *skipto( c, p, esc )
2 register char *p ;
3 register int c, esc ;
4 {
5     /*
6      * Skip to c or to end of string. If c is preceded by
7      * the esc character it is skipped. Return a pointer
8      * to c.
9     */
10
11    while( *p && *p != c )
12    {
13        if ( *p != esc )
14            p++ ;
15
16        else if ( *++p ) /* skip escaped characters */
17            p++ ;
18    }
19
20    return(p) ;
21 }

```

End Listing Eight

Listing Nine

```

1 uatoi(s)
2 char **s;
3 {
4     /* Like atoi but updates s to point past the number.
5      */
6
7     register char *str;
8     register int num = 0 ;
9
10    for( str = *s ; '0' <= *str && *str <= '9'; str++ )
11        num = (num * 10) + (*str - '0') ;
12
13    *s = str;
14    return num;
15 }
16
17 /***** -----
18
19 #ifdef DEBUG
20
21 #include <stdio.h>
22
23 main()
24 {
25     char buf[80], *bp;
26     int i;
27
28     while( 1 )
29     {
30         printf("enter string: ");
31         gets(buf);
32         bp = buf;
33         i = uatoi( &bp );
34         printf("num = %d, bp = <%s>\n", i, bp );
35     }
36 }
37
38 #endif

```

End Listings

Learn how to write shells applicable to MS-DOS, as well as to most other programming environments! This book and disk include a full description of a Unix-like shell, complete C source code, a thorough discussion of low-level DOS interfacing and significant examples of C programming at the system level.

Supported features include: • Read • Aliases • History • DOS compatible support • C-Shell-based shell scripts

The Unix-like control flow includes: if/then/else; while; foreach; switch/case; break; continue.

For IBM PC compatibles. All source code is included on disk.

/UTIL

When used with the shell described in *On Command*, these utility programs and subroutines provide a fully functional Unix-like environment! Utilities include: cat; cp; date; du; echo; grep; ls; mkdir; mv; p; pause; printenv; rm; rmdir; sub; and chmod. Complete source code and manual are included.

Receive *On Command* together with */UTIL* for only \$59.95!

| | |
|------------------------------|--------------------------|
| On Command | Item #163 \$39.95 |
| /UTIL | Item #161 \$29.95 |
| On Command with /UTIL | Item #164 \$59.95 |

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me **On Command: writing a Unix-Like Shell for MS-DOS** with disk for \$39.95

Send me **/UTIL** for \$29.95

Send me both **On Command** and **/UTIL** for only \$59.95

Subtotal _____

CA residents add sales tax _____ % _____

Add \$2.25 per item for shipping _____

TOTAL _____

Name _____

Address _____

City _____

State _____ Zip _____

Check Enclosed. Make payable to M&T Publishing.
Please Charge my VISA M/C AMEX

Card No. _____

Exp. Date _____

Signature _____

3124

STRUCTURED PROGRAMMING

Listing One (*Text begins on page 124.*)

Listing 1. Turbo Pascal listing of a four-function calculator program.

```
PROGRAM Calculate;
(* Four function calculator example *)

VAR OpError : BOOLEAN;
Operation, OK : CHAR;
X, Y, Result : REAL;

BEGIN
REPEAT
  ClrScr;
  WRITE('Enter first number '); READLN(X); WRITELN;
  WRITE('Enter operation '); READLN(Operation); WRITELN;
  WRITE('Enter second number '); READLN(Y); WRITELN;
  OpError := FALSE;
CASE Operation OF
  '+': Result := X + Y;
  '-': Result := X - Y;
  '*': Result := X * Y;
  '/': IF Y <> 0.
    THEN
      Result := X / Y
    ELSE BEGIN
      WRITELN('Cannot divide by zero!');
      WRITELN;
      OpError := TRUE
    END
  ELSE OpError := TRUE;
END; (* CASE *)
IF NOT OpError THEN BEGIN
  WRITELN(X,' ',Operation,' ',Y,' = ',Result);
  WRITELN;
END;
WRITE('Perform another operation? (Y/N) ');
READLN(OK); WRITELN;
UNTIL (OK <> 'Y') AND (OK <> 'y');
END.
```

End Listing One

Listing Two

Listing 2. Modula-2 listing of a four-function calculator program.

```
MODULE Calculate;

FROM ScreenHandler
IMPORT ClrEol, ClrScr, DelLine, InsLine, GotoXY, WhereX, WhereY,
CrtInit, CrtExit, LowVideo, NormVideo, HighVideo, SetAttribute,
GetAttribute, normalAtt, boldAtt, reverseAtt, underlineAtt,
blinkAtt, boldUnderlineAtt, blinkUnderlineAtt, boldBlinkAtt,
reverseBlinkAtt, boldUnderlineBlinkAtt;
FROM TRealIO IMPORT ReadReal, WriteReal;
FROM TTextIO
IMPORT ReadInt, ReadCard, ReadChar, ReadString, ReadLn, ReadBuffer,
WriteInt, WriteCard, WriteChar, WriteString, WriteBool, WriteLn,
Eoln, SeekEof, SeekEoln;
FROM TKernelIO
IMPORT File, FileType, OptionMode,
StatusProc, ReadProc, WriteProc, ErrorProc,
stdinout, input, output, con, trm, kbd, lst, aux, usr,
conStPtr, conInPtr, auxInPtr, usrInPtr, conOutPtr, lstOutPtr,
auxOutPtr, usrOutPtr, errorPtr, IOresult, KeyPressed, IOBuffer,
IOCheck, DeviceCheck, CtrlC, InputFileBuffer, OutputFileBuffer;

(* Four function calculator example *)

VAR
OpError: BOOLEAN;
Operation, OK: CHAR;
X, Y, Result: REAL;

BEGIN
REPEAT

  ClrScr;
  WriteString(stdinout, 'Enter first number ', 0);
  ReadBuffer(on);
  ReadReal(stdinout, X);
  ReadLn(stdinout);
  ReadBuffer(off);
```

(continued on page 72)

IS GETTING THE ANSWER TO SOFTWARE PROBLEMS A BIGGER PROBLEM THAN THE PROBLEM?

Don't stay on hold when there's help online from CompuServe® Software Forums.



The new upgraded version of your software locks up. And every time you reboot, you get stuck in the same place in the program.

You've chucked the manual, because you've done exactly what it tells you to do six times already. So you call the software company.

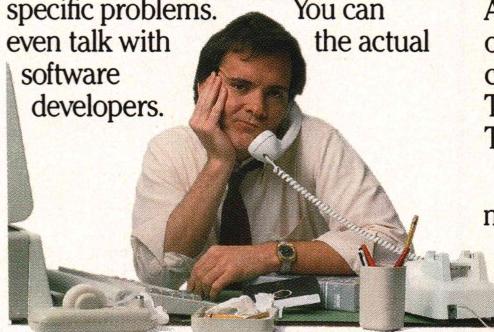
Now you spend half a day beating your head against a brick wall of busy signals, ranting at recorded messages, hanging around on hold. And you still don't get the solution to your problem.

Meanwhile, progress is stopped and your profits are dribbling away. But wait. There's help...

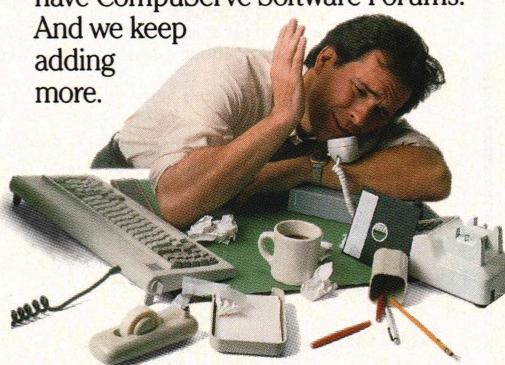
Several prominent, progressive software publishers recognize this problem, and working with CompuServe, have developed a solution—CompuServe Software Forums.

Now you can go online with experts from the companies that produced your software and get

prompt, written answers to your specific problems. You can even talk with the actual software developers.



Adobe Systems®, Aldus®, Ashton-Tate®, Autodesk®, Borland International®, Creative Solutions®, Digital Research®, Living Videotext®, Lotus® Inc., Microsoft®, MicroPro®, Misosys Inc.® and Software Publishing® all have CompuServe Software Forums. And we keep adding more.



CompuServe's large subscriber base also puts you in touch with thousands of other, often more experienced, users of the same software. You'll find they can give you lots of creative ways to get the most out of your software.

And software forums are the best way to learn about product updates, new product announcements, new ways to expand the uses of your software, and offer free uploads of your own programs.

Our online electronic magazines

frequently publish software reviews. And you can find help for many other software products in our other computer-related forums for IBM®, Tandy®, Atari®, Apple®, Commodore®, TI® and others.

The last thing you need when you've got a software problem is a bigger problem getting answers. So, from now on, get prompt, informed answers on CompuServe Software Forums.

To buy your CompuServe Subscription Kit, see your nearest computer dealer. Suggested retail price is \$39.95.

To order direct or for more information, call 800-848-8199 (in Ohio, 614-457-0802).

If you're already a CompuServe subscriber, just type GO SOFTWARE at any ! prompt.



CompuServe®

Information Services, P.O. Box 20212
5000 Arlington Centre Blvd., Columbus, OH 43220

An H&R Block Company

Circle no. 237 on reader service card.

Is Your Program HALF-FAST?

Speed it up
with



An execution profiler for IBM PCs and compatibles.

Most programs run at less than half the speed that they could. You can optimize almost any code, but where do you start?

A typical program spends 90% of its time in 10% of its code. **The Watcher** identifies that critical 10% for you, so you don't waste your effort on the wrong 90%.

The Watcher is easy to learn and easy to use, and we provide full technical support. **Watcher** users have increased program performance by as much as 300%. You can get similar results.

The Watcher works with any non-interpretive language on DOS version 2 or 3.

Turbo Pascal users: A special section of our manual is dedicated to you!



only
\$59.95



Plus \$3.00
Shipping &
Handling.

To order or for free information,
call or write:

Stony Brook
INC.
SOFTWARE

Forest Road
Wilton, New Hampshire 03086
(603) 654-2525

Circle no. 287 on reader service card.

STRUCTURED PROGRAMMING

Listing Two (Listing continued, text begins on page 124.)

```
WriteLn(stdinout);
WriteString(stdinout, 'Enter operation ', 0);
ReadBuffer(on);
ReadChar(stdinout, Operation);
ReadLn(stdinout);
ReadBuffer(off);
WriteLn(stdinout);
WriteString(stdinout, 'Enter second number ', 0);
ReadBuffer(on);
ReadReal(stdinout, Y);
ReadLn(stdinout);
ReadBuffer(off);
WriteLn(stdinout);
OpError := FALSE;
CASE Operation OF
  '+':
    Result := X+Y
  | '-':
    Result := X-Y
  | '*':
    Result := X*Y
  | '/':
    IF
      Y <> 0. THEN
        Result := X/Y
    ELSE
      WriteString(stdinout, 'Cannot divide by zero!', 0);
      WriteLn(stdinout);
      WriteLn(stdinout);
    END;
    OpError := TRUE;
  ELSE
    OpError := TRUE;
  END; (* CASE *)
IF NOT OpError THEN
  WriteReal(stdinout, X, 18, -10);
  WriteString(stdinout, ' ', 0);
  WriteChar(stdinout, Operation, 0);
  WriteString(stdinout, ' ', 0);
  WriteReal(stdinout, Y, 18, -10);
  WriteString(stdinout, ' = ', 0);
  WriteReal(stdinout, Result, 18, -10);
  WriteLn(stdinout);
  WriteLn(stdinout);
END;
WriteString(stdinout, 'Perform another operation? (Y/N)', 0);
ReadBuffer(on);
ReadChar(stdinout, OK);
ReadLn(stdinout);
ReadBuffer(off);
WriteLn(stdinout);
UNTIL (OK <> 'Y') AND (OK <> 'y');
END Calculate.
```

Listing Three

Listing 3. Turbo Pascal program for text pattern matching.

```
PROGRAM Pattern_Search_Test;
(*$V-*)
CONST MAX = 100;
  DEFAULT_LINE = 'Namir Clement Shammas';

TYPE STRING40 = STRING[40];
  STRING80 = STRING[80];
  STRING255 = STRING[255];

VAR Line : STRING255;
  Pattern : STRING40;

FUNCTION Pattern_Search(Text_Line : STRING255; Pattern : STRING40) : INTEGER;
(* Scan Text_Line with Pattern string, containing possible *)
(* combination of wildcards. *)

VAR Num_Tokens : INTEGER;
  Token : ARRAY [1..MAX] OF STRING40;
```

(continued on page 74)

"How to protect your software by letting people copy it!"

By Dick Erett, President of Software Security



Inventor and entrepreneur, Dick Erett, explains his company's view on the protection of intellectual property.

A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment

Hard Disk Installation : Simply copy program disk to hard disk using DOS Command - Copy A:.* C:

Program Back-ups : You may make as many copies of the program diskette as you wish.

Data Back-ups : Use normal back-up and restore commands, including backing up sub-directories containing program files.

User Area Networks : This product may be used on User Area Networks. Follow the same installation steps on page 102 of this manual. The Block will interfere with the normal operation of any

Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

of the market, or take a stand against the theft of your intellectual property.

... giving your software away is fine ...

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

... possibilities... limited only by your imagination...

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

**Software
Security inc.**

870 High Ridge Road Stamford, Connecticut 06905
203 329 8870

Circle no. 170 on reader service card.

C & PASCAL PROGRAMMERS

Blaise Computing provides a broad range of programming tools for Pascal and C programmers, with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

C TOOLS PLUS

\$175.00

NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

PASCAL TOOLS/TOOLS 2

\$175.00

Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

VIEW MANAGER

\$275.00

Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or field-by-field control. Specify C or IBM/MS-Pascal.

ASYNCH MANAGER

\$175.00

Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

Turbo POWER TOOLS PLUS

\$99.95

NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

Turbo ASYNCH PLUS

\$99.95

Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; and baud rates up to 9600.

RUNOFF

\$49.95

NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

EXEC

\$95.00

Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

ORDER TOLL-FREE 800-227-8087!



BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Circle no. 217 on reader service card.

STRUCTURED PROGRAMMING

Listing Three (Listing continued, text begins on page 124.)

```
PROCEDURE INC(VAR A : INTEGER);  
  
BEGIN  
  A := A + 1;  
END;  
  
FUNCTION Offset_Pos(Str1, Str2 : STRING80; Ptr : INTEGER) : INTEGER;  
  
VAR Ptr2 : INTEGER;  
  
BEGIN  
  Delete(Str1, 1, Ptr-1);  
  Ptr2 := Pos(Str2, Str1);  
  IF Ptr2 > 0 THEN Ptr2 := Ptr2 + Ptr - 1;  
  Offset_Pos := Ptr2;  
END; (* Offset_Pos *)  
  
PROCEDURE Scan_Pattern;  
  
VAR Char_Pos, Pattern_Length, Ptr, Ptr2 : INTEGER;  
  
BEGIN  
  Char_Pos := 1; Num_Tokens := 0; Ptr := 1;  
  Pattern_Length := Length(Pattern);  
  WHILE Char_Pos <= Pattern_Length DO BEGIN  
    CASE Pattern[Char_Pos] OF  
      '?' : BEGIN  
        IF Char_Pos > Ptr  
        THEN BEGIN  
          INC(Num_Tokens);  
          Token[Num_Tokens] :=  
            Copy(Pattern, Ptr, (Char_Pos - Ptr));  
        END; (* IF *)  
        Ptr2 := Char_Pos;  
        WHILE Pattern[Ptr2] = '?' DO INC(Ptr2);  
        INC(Num_Tokens);  
        Token[Num_Tokens] :=  
          Copy(Pattern, Char_Pos, (Ptr2 - Char_Pos));  
        Ptr := Ptr2; Char_Pos := Ptr2  
      END;  
      '*' : BEGIN  
        (* Resolve any pending strings *)  
        IF Char_Pos > Ptr  
        THEN BEGIN  
          INC(Num_Tokens);  
          Token[Num_Tokens] :=  
            Copy(Pattern, Ptr, (Char_Pos - Ptr));  
        END;  
        INC(Num_Tokens);  
        Token[Num_Tokens] := Pattern[Char_Pos];  
        Ptr := Char_Pos + 1;  
      END;  
      END; (* CASE *)  
      INC(Char_Pos)  
    END; (* WHILE *)  
    (* Store any trailing characters *)  
    IF Char_Pos > Ptr  
    THEN BEGIN  
      INC(Num_Tokens);  
      Token[Num_Tokens] := Copy(Pattern, Ptr, (Pattern_Length - Ptr + 1));  
    END;  
  END; (* Scan_Pattern *)  
  
FUNCTION Locate_Pattern : INTEGER;  
  
VAR I, First_Char, Ptr, Ptr2 : INTEGER;  
  
BEGIN  
  First_Char := 0; Ptr := 1; Ptr2 := 1; I := 1;  
  WHILE I <= Num_Tokens DO BEGIN  
    IF Pos('?', Token[I]) > 0  
    THEN BEGIN  
      (* Sub-pattern has one or more '?' *)  
      Ptr := Ptr + Length(Token[I]);  
      (* does the text following the '?..?' match ? *)  
      Ptr2 := Offset_Pos(Line, Token[I+1], Ptr);  
      IF Ptr <> Ptr2 THEN Ptr2 := 0;  
      INC(I);  
    END;  
  END;
```

(continued on page 78)

Your Quality Connection

When you need programmer's development tools, Programmer's Connection is your best one-stop source. We've specialized in development software for IBM personal computers since 1984 and are experienced in providing a full range of quality products and customer services.

Unbiased Advice. Our friendly, non-commissioned salespeople are always prepared to assist you. We also have experienced technical consultants who can answer questions, help you compare products and send you detailed product information tailored to your needs. Since we're not affiliated with any software publisher or manufacturer, we'll give you an unbiased look at the products we carry.

High Quality. We stock hundreds of high quality software development tools specifically for IBM personal computers and compatibles. And as new products become available, we'll sell only those that meet our high standards for quality and value.

Manufacturer Support. The products we sell are the latest versions and come with the same technical support as if buying directly from the manufacturer.

Return Guarantees. Our goal is customer satisfaction and that's why we offer a 30-day documentation evaluation period or a 30-day return guarantee on most of our products. Please call for specific details.

Immediate Shipment. Most products are in stock and are ready for shipment from our large inventory.

Discounts. You'll save money on all of your software purchases from Programmer's Connection. Our ads show both the discount and retail prices for each product so you'll always know exactly how much you'll save.

Credit Cards. We'll charge your credit card at the time we ship your order. Other companies may charge your credit card at the time they take your order so they can use your money interest-free while you wait for your shipment.

FREE Shipping. Shipping is FREE if you have your order shipped via standard UPS anywhere in the USA. We can also express your order to you with no special fees and we'll only charge you the shipping carrier's standard rate. Many other companies profit from overcharges plus special fees for express shipments.

No Sales Tax. Customers outside of Ohio are not charged state sales tax.

No Hidden Charges. Quite simply, the prices you see on the next two pages are all you pay. We don't charge extra for standard UPS shipping, credit cards, COD orders, purchase orders or special handling.

When you buy from Programmer's Connection, you get all of the benefits of buying directly from the manufacturer and none of the drawbacks. So call us today and discover the advantages of our one-stop service for yourself. You'll be glad you did!



Turn the page for our latest advertised price list and ordering information.

apl language

| | | |
|---|-----|-----|
| APL*PLUS/PC by STSC | 595 | 429 |
| APL*PLUS/PC Spreadsheet Mgr by STSC | 195 | 139 |
| APL*PLUS/PC Tools Vol 1 by STSC | 295 | 199 |
| APL*PLUS/PC Tools Vol 2 by STSC | 85 | 59 |
| APL*PLUS/UNIX For AT XENIX by STSC | 995 | 695 |
| Btrieve ISAM File Mgr by SoftCraft | 245 | 194 |
| Financial/Statistical Library by STSC | 275 | 195 |
| Pocket APL by STSC | 95 | 69 |
| STATGRAPHICS by STSC | 795 | 579 |

artificial intelligence

1st-Class by Programs in Motion

APT from Solution Systems

Arity Combination Package

Expert System Development Pkg

File Interchange Toolkit

PROLOG Compiler & Interpreter

Screen Design Toolkit

SUL Development Package

Arity PROLOG Interpreter

Arity Standard Prolog

AutoIntelligence by IntelligenceWare

ExpertEDGE Advanced by Human Edge

ExpertEDGE Professional by Human Edge

Expertech II by IntelligenceWare

EXSYS Development Software by EXSYS

EXSYS Runtime System

GCLISP Golden Common LISP by Gold Hill

GCLISP 286 Developer by Gold Hill

Insight 1 by Level Five Research

Insight 2+ by Level Five Research

Intelligence/Compiler IntelligenceWare

Logic-Line Series 1 by Thunderstone

Logic-Line Series 2 by Thunderstone

Logic-Line Series 3 by Thunderstone

LPA microPROLOG by Prag Logic Systems

with APES

LPA Professional microPROLOG

with APES

Microsoft LISP Common LISP

250 163

MPROLOG Language Primer LOGICWARE New

50 45

MPROLOG P500 by LOGICWARE

New 495 395

MPROLOG P550 by LOGICWARE

New 220 175

PC Scheme by Texas Instruments

95 85

Personal Consultant Easy by TI

495 439

Personal Consultant Plus by TI

2950 2599

Personal Consultant Runtime

95 85

PROLOG-2 Interpreter by ESI

450 419

PROLOG-2 Interpreter and Compiler

895 839

QNIAL by NIAL Systems

375 349

TransLISP from Solution Systems

95 CALL

Turbo PROLOG Compiler by Borland Int'l

100 65

assembly language

386 ASM/LINK Cross Asm by Phar Lap

495 CALL

8088 Assembler w/Z-80 Trans by 2500 AD

100 89

ASMLIB Function Library by BC Assoc

149 129

asmTREE B-Tree Dev System by BC Assoc

395 339

Cross Assemblers Various by 2500 AD

CALL CALL

Microsoft Macro Assembler

150 95

Norton Utilities by Peter Norton

100 59

Turbo EDITASM by Speedware

99 84

Uniware Cross Assemblers Various by SDS

295 249

Visible Computer: 8088 Software Masters

80 65

basic language

BetterBASIC by Summit Software

200 129

8087 Math Support

99 75

Btrieve Interface

99 75

C Interface

99 75

Run-time Module

250 169

EXIM Services Toolkit by EXIM

CALL CALL

Finally by Komputerworks

99 85

Inside Track from MicroHelp

65 51

MACH 2 by MicroHelp

75 61

Microsoft QuickBASIC

99 65

87 QB Pak by Hauppauge

New 69 59

Peeks 'Pokes from MicroHelp

45 37

Professional BASIC by Morgan

99 75

8087 Math Support

50 42

Stay-Res by MicroHelp

95 85

True Basic w/BASIC A Converter

200 99

True Basic w/Converter & Run-time

295 179

BASIC A Converter

50 45

Run-time Module

150 99

Various Other Utilities

50 45

Turbo BASIC by Borland Int'l

New 100 69

blaise products

ASYNC MANAGER Specify C or Pascal

Sale 175 119

C TOOLS PLUS

Sale 175 119

EXEC Program Chainer

Sale 95 65

LIGHT TOOLS for Datalight C

New Sale 100 69

PASCAL TOOLS

Sale 125 85

PASCAL TOOLS 2

Sale 100 65

PASCAL TOOLS & PASCAL TOOLS 2

Sale 175 99

RUNOFF Text Formatter

Sale 50 39

TURBO ASYNCH PLUS

Sale 100 69

TURBO POWER TOOLS PLUS

Sale 100 69

VIEW MANAGER Specify C or Pascal

Sale 275 165

borland products

EUREKA Equation Solver

New 100 69

REFLEX & REFLEX Workshop

200 129

REFLEX Data Base System

150 89

REFLEX Workshop

70 45

Turbo BASIC

New 100 69

Turbo DATABASE TOOLBOX

70 47

Turbo EDITOR TOOLBOX

70 47

Turbo GAMEWORKS TOOLBOX

70 47

Turbo GRAPHIX TOOLBOX

70 47

Turbo LIGHTNING

100 64

Turbo Numerical Methods Library

New 100 69

Turbo PASCAL and TUTOR

New 125 85

Turbo PASCAL with 8087 and BCD

100 64

Turbo TUTOR

40 28

Turbo Prolog Compiler

100 64

Word Wizard

70 47

Word Wizard and Turbo Lightning

150 94

c++

► C++ by Guidelines w/kernel 1.1

Sale 195 169

c compilers

► 68000/10/20 Cross Compiler by SDS

595 CALL

C86PLUS by Computer Innovations

New 497 CALL

Datalight C Compiler Small Model

Sale 60 45

Datalight Developer Kit

Sale 99 69

Datalight Optimum-C

New 139 99

► with LIGHT TOOLS Blaise

New, Sale 239 168

DeSmet C w/Debugger

159 138

DeSmet C w/Debugger & Large Case

209 184

Eco-C Development System by Ecosoft

125 83

Lattice C Compiler from Lattice

500 275

Mark Williams Let's C Combo Pack

New 125 99

Let's C Compiler

75 57

csd Source Level Debugger

75 57

Mark Williams MWC-86

495 289

► Microsoft C CodeView

Sale 450 265

► Wizard C Combo by Wizard Systems

Sale 750 529

► Wizard C Compiler

Sale 450 299

► ROM Development Pkg

Sale 350 259

c interpreters

► C-ter by Gimpel, Specify compiler

Sale 300 199

C Trainer with Book by Catalyxit

122 87

► Instant C by Rational Systems

Sale 500 359

► Introducing C by Computer Innovations

125 CALL

Run/C from Lifeboat

150 89

Run/C Professional from Lifeboat

250 169

c utilities

► APT by Shaw American Technology

Sale 395 259

Basic C Library by C Source

175 119

► C Essentials by Essential Software

Sale 100 65

C-ISAM by Informix

225 195

C to dBase by Computer Innovations

150 CALL

c-tree & r-tree Combo by FairCom

Sale 650 459

c-tree ISAM File Manager

Sale 395 279

c-tree Report Generator

Sale 295 199

► C Utility Library by Essential

Sale 185 119

C Windows by Syscom

100 85

► Superfronts for C

50 43

► Essential Comm Library w/Debugger

Sale 250 159

► Breakout Debugger Any language

Sale

logitech products

| | | | |
|---|------------------------------|-----|----|
| LOGIMOUSE C7 | Specify Connector Type . . . | 99 | 83 |
| with PLUS Pkg | 119 | 98 | |
| with PLUS Pkg & PC Paintbrush | 169 | 134 | |
| with PLUS Pkg & CAD Software | 189 | 153 | |
| with PLUS Pkg & CAD & Paint | 219 | 179 | |
| MODULA-2/86 Holiday Package | 199 | 159 | |
| MODULA-2/86 Compiler | 89 | 62 | |
| MODULA-2/86 with 8087 Support | 129 | 98 | |
| MODULA-2/86 with PLUS Pkg | 189 | 138 | |
| Library Sources | 99 | 88 | |
| Make Utility | 29 | 25 | |
| ROM Package | 199 | 172 | |
| Run Time Debugger | 69 | 56 | |
| Turbo to Modula Translator | 49 | 42 | |
| Utilities Package | 49 | 42 | |
| Window Package | 49 | 42 | |
| REPERTOIRE for MODULA-2/86 by PMI | 89 | 79 | |
| Object Code Only | New | 19 | |
| | | 15 | |

micropoint products

| | | | |
|---|------|-----|-----|
| ► System V/AT by Micropoint Systems | Sale | 440 | 359 |
| Runtime System (Operating System) | 159 | 145 | |
| Software Development System | 169 | 155 | |
| Text Preparation System | 169 | 155 | |
| User Upgrade 3 to Unlimited Users | 169 | 155 | |

microsoft products

| | | | |
|---|------|------|-----|
| Microsoft BASIC for XENIX | 350 | 239 | |
| ► Microsoft C with CodeView | Sale | 450 | 265 |
| Microsoft COBOL Compiler | 700 | 439 | |
| for XENIX | 995 | 635 | |
| Microsoft COBOL Tools with Debugger | 350 | CALL | |
| for XENIX | 450 | 289 | |
| Microsoft FORTRAN Compiler | 350 | 204 | |
| for XENIX | 695 | 439 | |
| Microsoft Learning DOS | 50 | 36 | |
| Microsoft LISP Common LISP | 250 | 163 | |
| Microsoft MACH 10 w/Mouse & Windows | 549 | 385 | |
| Microsoft MACH 10 Board only | 399 | 285 | |
| Microsoft Macro Assembler | 150 | 95 | |
| Microsoft Mouse Bus Version | 175 | 114 | |
| Microsoft Mouse Serial Version | 195 | 124 | |
| Microsoft muMath Includes muSIMP | 300 | 184 | |
| Microsoft Pascal Compiler | 300 | 184 | |
| for XENIX | 695 | 439 | |
| Microsoft QuickBASIC | 99 | 65 | |
| Microsoft Sort | 195 | 129 | |
| Microsoft Windows | 99 | 65 | |
| Microsoft Windows Development Kit | 500 | 309 | |

other languages

| | | |
|---|-----|-----|
| CCS MUMPS Single-User by MGlobal | 60 | 51 |
| CCS MUMPS Single-User/Multi-Tasking | 150 | 129 |
| CCS MUMPS Multi-User | 450 | 369 |
| Janus/ADA C Pack by R&R Software | 95 | 89 |
| Janus/ADA D Pack by R&R Software | 900 | 769 |
| Methods Smalltalk by Digitalink | 79 | 66 |
| Personal REXX by Mansfield Software | 125 | 99 |
| Smalltalk/V by Digitalink | 99 | 84 |
| Smalltalk/Comm | 49 | 45 |
| SNOBOL4+ by Catspaw | 95 | 80 |

other products

| | | |
|---|------|------|
| Compact Source Print by Aldebaran | New | CALL |
| Dan Bricklin's Demo Pgm Software Garden | 75 | 59 |
| FANSI-CONSOLE by Hersey Micro | New | 75 |
| FASTBACK by 5th Generation Systems | 179 | 149 |
| Informix for DOS by Informix | 795 | 639 |
| Informix4GL for DOS by Informix | 995 | 789 |
| InformixSQL for DOS by Informix | 795 | 639 |
| Instant Replay by Nostradamus | 90 | 79 |
| Interactive EASYFLOW by Haventree | 150 | 129 |
| MKS Toolkit with vi by MKS | 139 | 119 |
| Norton Commander by Peter Norton | 75 | 55 |
| OPT-Tech Sort by Opt-Tech Data Proc | 149 | 115 |
| PrintOut by Software Directions | 89 | 84 |
| Quilt Computing Combo Package | 199 | 169 |
| QMake Program Rebuild Utility | 99 | 84 |
| SRMS Software Revision Mgmt Sys | 125 | 109 |
| screenplay all varieties by Flexus | CALL | CALL |
| SoftScreen/HELP by Dialectic Systems | 195 | 149 |
| Source Print by Aldebaran Labs | 97 | CALL |
| Taskview by Sunny Hill Software | 80 | 56 |
| TLIB by Burton Systems Software | 100 | 89 |
| Tree Diagrammer by Aldebaran Labs | New | CALL |
| VTEK Term Emulator by Sci Endeavors | 150 | 129 |

phoenix products

| | | |
|--|------|-----|
| Pasm86 Macro Assembler Version 2.0 | 195 | 115 |
| Pdisk Hard Disk & Backup Utility | 195 | 125 |
| Pfantasy Pac Phoenix Combo | 1295 | 849 |
| Pfinish Performance Analyzer | 395 | 229 |
| Pfix-86 Plus Symbolic Debugger | 395 | 229 |

LOWEST PRICES

Since this ad is prepared in advance of publication, some of our current prices may be lower than what's advertised here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and telephone number.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. POs with net 30-day terms are available to qualified US accounts only.

FOREIGN ORDERS

Shipping charges for foreign and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. Foreign orders (except Canada), please include an additional \$10 for customs form preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Call for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can assist in comparing products, answer technical questions and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products come with a 30-day documentation evaluation period or 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

CALL TOLL-FREE

| | |
|------------------|-----------------------------|
| U S | 800-336-1166 |
| CANADA | 800-225-1166 |
| OHIO & ALASKA | (Call Collect) 216-877-3781 |
| FOREIGN | 216-877-3781 |
| CUSTOMER SERVICE | 216-877-1110 |

Hours: Weekdays 8:30 AM to 8:00 PM EST.

Ohio customers add 5% state sales tax.

136 SUNNYSIDE ST. HARTVILLE, OHIO 44632

| | | |
|---|-----|-----|
| PforCe Comprehensive C Library | 395 | 229 |
| Plink-86 Plus Overlay Linker | 495 | 319 |
| Pmaker Make Utility | 125 | 78 |
| Pmate Macro Text Editor | 195 | 115 |
| Pre-C Lint Utility | 295 | 155 |
| Ptel Binary File Transfer Program | 195 | 115 |

polytron products

| | | |
|---|------|------|
| PolyBoost The Software Accelerator | 80 | 69 |
| Polytron C Beautifier | 49 | 45 |
| Polytron C Library I | 99 | 78 |
| Polytron PowerCom Communications | 179 | 139 |
| PolyLibrarian Library Manager | 99 | 78 |
| PolyLibrarian II Library Manager | 149 | 115 |
| PolyMake UNIX-like Make Facility | 99 | 78 |
| PolyShell | 149 | 119 |
| PolyWindows Products All Varieties | CALL | CALL |
| PolyXREF Complete Cross Ref Utility | 125 | 99 |
| PolyXREF One language only | 129 | 99 |
| PVCS Version Control System | 395 | 309 |

softcraft products

| | | |
|--|-----|-----|
| Btrieve ISAM Mgr with No Royalties | 245 | 194 |
| Xtrieve Query Utility | 245 | 194 |
| Report Option | 145 | 114 |
| Btrieve/N for Networks | 595 | 464 |
| Xtrieve/N | 595 | 464 |
| Report Option/N | 345 | 274 |

text editors

| | | |
|---|-----|------|
| Brief from Solution Systems | 195 | CALL |
| Epsilon Emacs-like editor by Lugaru | 195 | 159 |
| KEDIT by Mansfield Software | 125 | 99 |
| PC/VI by Custom Software Systems | 149 | 119 |
| SPF/PC by Command Technology Corp. | 195 | 139 |
| Vedit by CompuView | 150 | 107 |
| Vedit Plus by CompuView | 185 | 139 |

turbo pascal utilities

| | | |
|--|-----|------|
| ALICE Interpreter by Software Channels | 95 | 66 |
| Flash-up Windows by Software Bottling | 90 | CALL |
| HELP/Control by MDS | 125 | 109 |
| On-line Help from Opt-Tech Data Proc | 149 | 109 |
| Report Builder by Royal American | 75 | CALL |
| Screen Sculptor by Software Bottling | 125 | 91 |
| System Builder by Royal American | 100 | CALL |
| TDebugPLUS by TurboPower Software | 60 | 49 |
| Turbo EXTENDER by TurboPower Software | 85 | 64 |
| Turbo Professional by Sunny Hill | 70 | 48 |
| TurboHALO from IMSI | 129 | 98 |
| TurboPower Utilities by TurboPower | 95 | 78 |
| TurboRef by Gracan Services | 50 | 45 |
| TurboSmith Visual Age Debugger | 69 | 45 |
| TurboWINDOW by MetaGraphics | 80 | 65 |

wendin products

| | | |
|--|----|----|
| Operating System Toolbox | 99 | 79 |
| PCNX Operating system | 99 | 79 |
| PCVMS Similar to VAX/VMS | 99 | 79 |
| XTC Text editor with Pascal source | 99 | 79 |

xenix system v

| | | |
|---|------|-----|
| See also Micropoint System V/AT section. | | |
| XENIX System V Complete by SCO | 1295 | 999 |
| XENIX Development System | 595 | 499 |
| XENIX Operating Sys Specify XT/AT | 595 | 499 |
| XENIX Text Processing Package | 195 | 144 |

xenix products

| | | |
|---|------|------|
| Btrieve ISAM File Mgr by SoftCraft | 595 | 464 |
| C-ISAM by Informix | 319 | 285 |
| c-tree ISAM Mgr by FairCom | 395 | 279 |
| dbVISTA All Varieties by Raima | Sale | CALL |
| dbVista with Library Source by Desktop AI | Sale | CALL |
| DOSIX User Version by Data Basics | New | 199 |
| DOSIX Console Version by Data Basics | New | 399 |
| Informix by Informix | 995 | 795 |
| Informix4GL by Informix | 1500 | 1239 |
| InformixSQL by Informix | 995 | 795 |
| Lyrix by Informix | 595 | 449 |
| Micro Focus Level II Compact COBOL | 1000 | 795 |
| Forms-2 | 400 | 319 |
| Level II ANIMATOR | 600 | 479 |
| Microsoft See Microsoft Section | CALL | CALL |
| Networks for XENIX by SCO | 595 | 495 |
| PANEL Screen Designer by Roundhill | 625 | 535 |
| REAL-TOOLS Binary Version by PCT | New | 149 |
| Library Source Version | New | 399 |
| Complete Source Version | New | 499 |
| RM/COBOL by Ryan-McFarland | 1250 | 949 |
| RM/FORTRAN by Ryan-McFarland | 750 | 549 |
| SCO Professional Lotus clone by SCO | 795 | 595 |

STRUCTURED PROGRAMMING

SAPIENS V8
A VIRTUAL MEMORY MANAGER
FOR THE PC

C PROGRAMMERS!
LINK IN

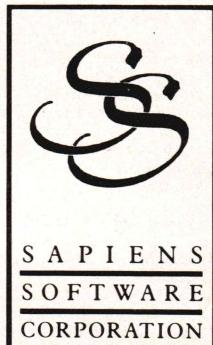
EXPAND YOUR C COMPILER
8 MEGABYTES
ON A PC

- ★ VIRTUAL MEMORY MANAGER FOR C PROGRAMMERS ON THE IBM PC.
- ★ PROVIDES 8 MGS. VIRTUAL MEMORY WORKSPACE.
- ★ Link V8 libraries to C compilers — MICROSOFT, LATTICE AND AZTEC.
- ★ FAST: LESS THAN 10% SPEED OVERHEAD
- ★ ADVANCED SOFTWARE EMULATION OF 64-BIT ARCHITECTURE

\$300⁰⁰

Sapiens V8 is for C programmers who want to develop PC applications that go beyond current memory restrictions of the PC and a 16 bit architecture.

- V8 is not dependent on add-on boards.
- Virtual stack library supports stack frame management and multiple return values.
- Virtual heap (vmalloc()) allows allocation of very large data structures.
- System requirements: Huge model C compiler, V8 uses 22-128 Kb core.



Sapiens Software Corporation
236 Mora St. Santa Cruz, CA 95060
408/458-1990

Circle no. 168 on reader service card.

Listing Three (Listing continued, text begins on page 124.)

```

IF (Pos('?',Token[I]) > 0) OR (Token[I] <> '**')
THEN Ptr2 := Offset_Pos(Line, Token[I], Ptr);

IF (Token[I] <> '**')
THEN BEGIN
  IF (Ptr2 = 0) AND (First_Char > 0)
  THEN BEGIN
    First_Char := 0;
    I := I - 1
  END
  ELSE
    IF (Ptr2 = 0) AND (First_Char = 0)
    THEN
      I := Num_Tokens
    ELSE BEGIN
      IF (First_Char = 0) THEN First_Char := Ptr2;
      Ptr := Ptr2 + Length(Token[I]);
    END;
  END;
  INC(I);
END; (* WHILE I *)
Locate_Pattern := First_Char;
END; (* Locate_Pattern *)

BEGIN (* Pattern_Search *)
  Scan_Pattern;
  Pattern_Search := Locate_Pattern
END; (* Pattern_Search *)

BEGIN (*----- M A I N -----*)
  ClrScr;
  WRITELN('Default string is : ',DEFAULT_LINE); WRITELN;
  WRITE('Enter string '); READLN(Line); WRITELN;
  IF Line = '' THEN Line := DEFAULT_LINE;
  WRITE('Enter search pattern string '); READLN(Pattern); WRITELN;
  WRITELN('Matches at position ',Pattern_Search(Line,Pattern));
  WRITELN;
END.

```

End Listing Three

Listing Four

Listing 4. Modula-2 program for text pattern matching.

```

MODULE PatternSearchTest;

FROM Strings
  IMPORT Assign, Insert, Delete, Pos, Copy, Concat, Length, CompareStr;
FROM ScreenHandler
  IMPORT ClrEol, ClrScr, DelLine, InsLine, GotoXY, WhereX, WhereY,
  CrtInit, CrtExit, LowVideo, NormVideo, HighVideo, SetAttribute,
  GetAttribute, normalAtt, boldAtt, reverseAtt, underlineAtt,
  blinkAtt, boldUnderlineAtt, blinkUnderlineAtt, boldBlinkAtt,
  reverseBlinkAtt, boldUnderlineBlinkAtt;
FROM TTextIO
  IMPORT ReadInt, ReadCard, ReadChar, ReadString, ReadLn, ReadBuffer,
  WriteInt, WriteCard, WriteChar, WriteString, WriteBool, WriteLn,
  Eoln, SeekEof, SeekEoln;
FROM TKernelIO
  IMPORT File, FileType, OptionMode,
  StatusProc, ReadProc, WriteProc, ErrorProc,
  stdInout, input, output, con, trm, kbd, lst, aux, usr,
  constPtr, conInPtr, auxInPtr, usrInPtr, conOutPtr, lstOutPtr,
  auxOutPtr, usrOutPtr, errorPtr, IOresult, KeyPressed, IOBuffer,
  IOCheck, DeviceCheck, CtrlC, InputFileBuffer, OutputFileBuffer;
(*-----*)
(* Copyright (c) 1986, Namir Clement Shammas *)
(*-----*)

(* Compiler Directive not supported (*$V-* *) *)
(* See the MODULA-2 feature 'Open Array Parameters' *)

CONST
  MAX = 100;
  HI = 32000; (*--> line added *)

TYPE
  STRING40 = ARRAY [0..40-1] OF CHAR;

```

(continued on page 80)

We Do Windows!



WARNING: This product may promote large incomes, high productivity, and excessive free time.

Magus, Inc. is proud to present Data&Windows: a window-oriented data-entry system based on a new, revolutionary design philosophy. The only problem is... what should we call it?

It is easy to learn and use, like a **panel generator**, because it allows you to draw your text, fields, and colors on the video display. But we can't call it a panel generator, because it supports full windowing and scrolling, and because screen and field information may be stored in your program files (.EXE) rather than separate data files.

It is flexible and powerful, like a **library-oriented programmer's toolkit**, but you are not restricted to "visualizing" your data-entry windows as you type page after page of code to set up borders, fields, text and highlighting. Our innovative approach (called **static windowing**) eliminates the need for replication of static data in dynamic memory.

It produces tight code, like a **YACC** (Yet Another Compiler Compiler), but you don't have to tolerate a myriad of small program modules that need to be compiled and maintained. Instead, our "screen designer" creates Microsoft object files which you simply link with your applications.

Add to this new, superior design philosophy the fact that it has more features, produces tighter code, and yields higher performance than any of the above. Throw in a clear, concise user manual, a thorough on-disk tutorial, and some example programs. Top it off with a utility program that documents each screen and another that allows you to prototype (or simulate) your application before you write a single line of code. Now, what would *you* call it?

Let's settle on a single word.
Let's call it the "best."

But don't take our word for it. Order your demo disk today. You will receive a copy of the screen generator, the tutorial, and some documentation on the utility programs and library routines. Then make the decision yourself.

Or take advantage of our one-time introductory offer and get \$100 discount if you order before March 31, 1987.

Call (713) 665-4109 for more information.
Major credit cards accepted.

SCREEN DESIGNER:

Move/delete/center/fill/highlight block, global field redefinition, move/resize window and buffer, add/modify/move/delete field, insert/delete/undelete line, test/save/abort window, graphics characters, paint, jump-to-field, many cursor movement options, monitor switching, operating system calls, help. Set validation mode, highlight current field, scrolling by line or page, input mask, tabs, initial values. More. Up to 400 lines per screen.

FIELD DEFINITION:

Left-justify/right-justify/center, uppercase translation, built-in character validation, byte/integer/word/long/float/double/string/date field validation, retain data, auto-erase, protected fields, input required, use commas, use zeros/spaces, margin bell. User-defined character validations, pattern-matching validations, picture validations, and field types. More. Up to 9999 fields per screen.

LIBRARY ROUTINES:

Open, close, move, display, and refresh windows. Allow user to edit data fields in window, or to view and manipulate a window but not change data stored in it. Pull-down and pop-up menus. Read screen object file from disk. Intercept keyboard filter. Override default key actions. Automatic and manual refresh. Switch display device, erase all data fields on window, plot data onto fields or entire screens, retrieve data from fields or entire screens, screen image dump, retrieve and modify screen and field attributes, locate field, force use of bios. Direct interfacing with some bios interrupts, including cursor and mouse control. More. Mnemonic and simple to use.

REQUIREMENTS:

IBM PC/XT/AT/JR or true compatible, DOS 2.0 or later, at least 128K free RAM, and the Microsoft C, Pascal, or Fortran compiler or the IBM C compiler. Support is available for other C Compilers and the XENIX operating system. Call for specifics.

IBM, IBM PC, IBM XT, and IBM AT are trademarks of International Business Machines. Microsoft and XENIX are trademarks of Microsoft Corporation.

Circle no. 336 on reader service card.

For More Information

(713) 665-4109



Magus Inc.

Screen Designer

Let's Do Windows!

| | | |
|------------------------------|---------------------------------|-----------------------------------|
| <input type="checkbox"/> C | <input type="checkbox"/> Pascal | <input type="checkbox"/> FORTTRAN |
| Please send ___ copies @ | | \$345.00 |
| Introductory discount | | -100.00 |
| Your price | | 245.00 |

| | | |
|--|--|----------------|
| <input type="checkbox"/> Data&Windows with Library | | |
| Source Code | | \$695.00 |
| Introductory discount | | -100.00 |
| Your price | | 595.00 |

Hurry! Introductory offer expires March 31, 1987.

Show Me More!

| | |
|---|---------|
| <input type="checkbox"/> Send me a Demo | \$10.00 |
|---|---------|

In Texas add 6.125% sales tax _____
Outside U.S. add 15.00 _____
Total enclosed \$ _____

Enclosed is
 Check Money Order
 Visa MasterCard
Number _____
Expiration Date _____

Name _____
Company _____
Address _____
City _____
State _____ Zip Code _____

Send to:

MAGUS, INC.
4545 Bissonet Suite #114
Bellaire, TX 77401

ANNOUNCING . . . High performance APL Interpreter using MC68000 32 bit coprocessors and NS32081 floating point processors totally integrated with DOS and Novell Netware hardware and software environment. MultiAPL coprocessors offer 1MB or 4MB RAM for large APL workspaces and the fastest processing facilities available under DOS or Netware.

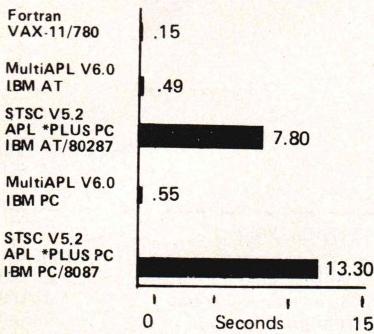
MULTIAPL

EXPLORE

- 640K + 1MB or 4MB
- Full component multi-user file system
- Btrieve file interface
- No restrictions on object size
- Shared variable interface
- Uses standard DOS files
- Overlays (functions/variables)
- 10 & 12 MHZ coprocessors available
- Extended superset of IBM's VSAPL
- APL*PLUS conversion utilities
- Full screen facilities included
- Enhanced version of APL.68000
- Run time versions available

COMPARE

BYTE Magazine
Calculations Benchmark
Double Precision Numbers
(All systems with one user)



INTRODUCTORY OFFER

\$995

GOOD THRU 4/30/87

INCLUDES: APL INTERPRETER AND REFERENCE MANUAL, 10 MHZ COPROCESSOR WITH 1 MB RAM (No Wait State)

Optional Math Processor \$295

Order direct for \$995 + Shipping/handling (\$18 US, \$20 Canada) VISA/MC/AMEX add 4%. Certified check, MO, COD. OFFER GOOD IN U.S. AND CANADA ONLY.

30 DAY MONEY BACK GUARANTEE

SPENCER
ORGANIZATION, INC.

P.O. BOX 248 WESTWOOD, N.J. 07675

(201) 666-6011

Btrieve is a trademark of Softcraft, Inc.
APL*PLUS is a trademark and service mark of STSC, Inc.

STRUCTURED PROGRAMMING

Listing Four (Listing continued, text begins on page 34.)

```
STRING80 = ARRAY [0..80-1] OF CHAR;
STRING255 = ARRAY [0..255-1] OF CHAR;
```

VAR

```
Line, DEFAULTLINE: STRING255;
Pattern: STRING40;
```

```
(* Function to scan Text_Line with Pattern string, containing possible *)
(* combination of wildcards. *)
```

```
PROCEDURE PatternSearch(TextLine: (*--> STRING255 *) ARRAY OF CHAR;
Pattern: (*--> STRING40 *) ARRAY OF CHAR): INTEGER;
```

VAR

```
NumTokens: INTEGER;
Token: ARRAY [1..MAX] OF STRING40;
```

```
(*--> PROCEDURE INC was removed *)
```

```
PROCEDURE OffsetPos(Str1, Str2: (* STRING80 *) ARRAY OF CHAR;
Ptr: INTEGER): INTEGER;
```

VAR

```
Ptr2: INTEGER;
```

```
VAR OffsetPosResult: INTEGER;
```

BEGIN

```
(*--> Delete(Str1, 1, Ptr-1); *)
IF Ptr > 0 THEN Delete(Str1, 0, Ptr-1) END;
Ptr2 := Pos(Str2, Str1);
(*--> IF Ptr2 > 0 THEN *)
IF CARDINAL(Ptr2) <= HIGH(Str1) THEN
  Ptr2 := Ptr2 + Ptr - 1;
ELSE (*--> ELSE clause added *)
  Ptr2 := HI;
END;
OffsetPosResult := Ptr2;
RETURN OffsetPosResult
END OffsetPos; (* Offset_Pos *)
```

```
PROCEDURE ScanPattern;
```

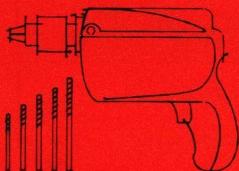
VAR

```
CharPos, PatternLength, Ptr, Ptr2: INTEGER;
```

BEGIN

```
(*--> CharPos := 1; *) CharPos := 0;
NumTokens := 0;
(*--> Ptr := 1; *) Ptr := 0;
PatternLength := Length(Pattern);
(*--> WHILE CharPos <= PatternLength DO *)
WHILE CharPos < PatternLength DO
  CASE Pattern[CharPos] OF
    '?':
      IF
        CharPos > Ptr THEN
          INC(NumTokens);
          Copy(Pattern, Ptr, (CharPos-Ptr), Token[NumTokens]);
      END; (*--> IF *)
      Ptr2 := CharPos;
      WHILE Pattern[Ptr2] = '?' DO
        INC(Ptr2)
      END;
      INC(NumTokens);
      Copy(Pattern, CharPos, (Ptr2-CharPos), Token[NumTokens]);
      Ptr := Ptr2;
    '*':
      CharPos := Ptr2
  | '*':
    (* Resolve any pending strings *)
    IF
      CharPos > Ptr THEN
        INC(NumTokens);
```

(continued on page 82)



POWER TOOLS for SYSTEM BUILDERS™

800-543-6277

Ask for Operator 2053
California: 800-368-7600

TSF is owned and operated by programmers, so we understand your needs. We believe and practice integrity in all business dealings. We advertise real prices and offer our best price to all customers. We provide prompt delivery of current product versions.

We accept checks, Visa, MasterCard and American Express. We charge your card only when we ship and do not add a surcharge. Free UPS delivery on orders over \$100. We allow return privileges on most products. **Give us a try!**

The Software Family

649 Mission Street
San Francisco, CA 94105
(415) 957-0111



Basic: *Mach2* from *MicroHelp* expands Basic's horizons with window management, input editing, array sorting and print formatting. It also provides MSDOS/BIOS function calls and allows you to break the 64K data barrier. Written in assembler for high performance. For Bascom, Quick Basic, BASIC-A and GWBASIC. (List \$75) Only \$60 from TSF. 30 day money back guarantee.

Every craftsman has a box filled with his or her personal collection of tools. Knowing which tool to use and how to use it can mean the difference between a quality product and a disaster. **TSF** has the **power tools** you need to produce quality software.

We suggest the following additions to your toolbox ...

C: *Gimpel's PC-Lint* evaluates your source code to locate insidious C bugs like "=" instead of "==" and mismatched function parameters. Evaluates multiple source files to validate use of all public functions and variables. Compatible with all popular compilers. 30 day trial period (15% restock fee). (List \$139) Only \$103 from **TSF**.

All Languages: *Periscope II* speeds debugging with multiple windows, symbolic memory addressing, and sophisticated trace and breakpoints. Includes an NMI break switch so you can interrupt programs at any time without pre-setting break points -- ends guesswork for hard-to-locate crashes. (List \$145) Only \$108 from **TSF**. 30 day money back guarantee!

Turbo: *Kyodor's Symbolic Debugger* provides symbolic debugging for Turbo Pascal programs. Includes trace, breakpoint, memory display and memory modification. Runs from within Turbo's environment, so you don't lose any Turbo features. Written in assembler for speedy operation and low memory overhead. (List \$49) Only \$40 from **TSF**.

Communications: *RamNet* from *System Design* provides background file transfers and e-mail for your PC. Incoming and outgoing calls are processed automatically while you continue with your normal work. Outgoing calls can be scheduled to take advantage of late night phone rates and to concentrate messages in a multi-node *RamNet* network. Only \$80 from **TSF**. A two node starter kit is only \$140. 30 day money back guarantee!

Winter Specials

Basic Language

| | |
|-----------------------------------|-------|
| Microsoft Quick Basic (List \$99) | \$65 |
| Summit Better Basic (List \$199) | \$160 |
| MicroHelp Mach 2 (List \$75) | \$60 |
| MicroHelp Stay-Res (List \$95) | \$75 |
| Sterling Castle Tools (List \$99) | \$80 |

Pascal Language

| | |
|-----------------------------------|-------|
| Microsoft Pascal (List \$300) | \$219 |
| Blaise View Manger (List \$275) | \$197 |
| Blaise Async Manager (\$175) | \$125 |
| Blaise Pascal Tools+ (List \$175) | \$125 |

Turbo Pascal

| | |
|----------------------------------|------|
| Turbo Pascal BCD & 8087 (\$100) | \$80 |
| Software Channels Alice (\$95) | \$85 |
| Kyodor Symbolic Debugger (\$49) | \$40 |
| Blaise Turbo Async (\$100) | \$80 |
| Blaise Power Tools (\$100) | \$80 |
| Borland Turbo Editor (List \$70) | \$50 |

Support Tools

| | |
|---------------------------------|-------|
| Custom Software PC/VI (\$149) | \$119 |
| MKS Toolkit (\$List \$139) | \$120 |
| Periscope I (List \$295) | \$225 |
| Periscope II (List \$145) | \$108 |
| Phoenix Plink+ (List \$495) | \$325 |
| Phoenix Pfinish (List \$395) | \$275 |
| Phoenix Pfix+ (List \$395) | \$250 |
| Polytron Make (List \$99) | \$80 |
| Quilt SRMS + Qmake (List \$199) | \$165 |
| Seidl SVM + SMK (List \$379) | \$330 |

C Language

| | |
|----------------------------------|-------|
| Datalight C (List \$99) | \$75 |
| Microsoft C w/Codeview (\$450) | \$285 |
| Lattice C (List \$500) | \$289 |
| Mark Williams C (List \$500) | \$319 |
| Rational Instant C | \$375 |
| Gimple C-Terp (List \$300) | \$224 |
| Run/C Professional (List \$250) | \$160 |
| Run/C Interpreter (List \$120) | \$85 |
| Gimpel PC Lint (List \$139) | \$103 |
| Blaise View Manager (List \$275) | \$197 |
| Lattice Curses (List \$125) | \$90 |
| Essentials Graphics (List \$250) | \$200 |
| Blaise C Tools+ (List \$175) | \$125 |
| Essentials Utility (List \$185) | \$130 |
| Greenleaf Functions (List \$185) | \$130 |
| Greenleaf Comm (List \$185) | \$130 |
| Blaise Async Manager (\$175) | \$125 |

... and many more

... call for price list

built
computing

MICROSOFT

BORLAND
INTERNATIONAL

Lattice

Phoenix

BLAISE COMPUTING

Circle no. 230 on reader service card.

C Users' Group

**A comprehensive, indexed
Directory of Public Domain
C Source Code is now available.**

To order, send \$10 (U.S.) to:

The C Users' Group

Post Office Box 97
McPherson, KS 67460
(316) 241-1065

Write or call for more information on
CUG membership.

Circle no. 181 on reader service card.

SOURCE CODE LIBRARIAN & REVISION CONTROL SYSTEM

TLIB™ keeps ALL versions of your program in ONE compact library file, even with hundreds of revisions!

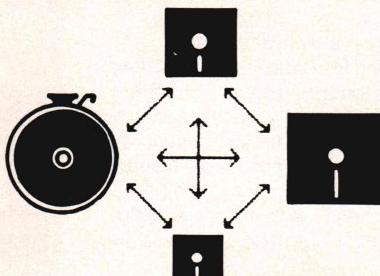
- Super Fast! Updates (deltas) average 5-7 times faster than PC/IX (Unix) SCCS. TLIB updates libraries faster than many editors load and save files!
- LAN-compatible! Shared libraries with PC Network!
- Synchronized control of multiple related source files.
- Use with floppies or hard disk. TLIB doesn't need big temporary files, so you can maintain a 300K library on one 360K diskette, with room to spare, even with TLIB itself on the same disk. And libraries are more compact than with most other revision management systems.
- Perfect for backup. Date and comments kept with each version. High data integrity because library data, once written, is never modified. Libraries are only appended, to minimize the chance of data loss due to a power glitch or hardware crash. And TLIB isn't copy-protected, either.
- Free copy of Landon Dyer's excellent public domain MAKE utility. With macros, full source code. For DOS & VAX/VMS. PC/MS-DOS 2.x & 3.x. Just \$99.95 + \$3 s/h Visa/MC

BURTON SYSTEMS SOFTWARE

P. O. Box 4156, Cary, NC 27511-4156
(919) 469-3068

Circle no. 212 on reader service card.

DATA CONVERSION



TRANSFER DATA BETWEEN OVER
600 DIFFERENT COMPUTER SYSTEMS

WORD PROCESSORS TOO

QUICK TURN-AROUND

PRICES FROM \$9 PER DISK

CALL OR WRITE FOR YOUR

FREE CATALOG

PORT-A-SOFT

555 S. STATE ST., SUITE #12
P.O. BOX 1685, OREM, UT 84057
(801) 226-6704

Circle no. 229 on reader service card.

STRUCTURED PROGRAMMING

Listing Four (Listing continued, text begins on page 124.)

```

        Copy(Pattern, Ptr, (CharPos-Ptr), Token[NumTokens]);
END;
INC(NumTokens);
Token[NumTokens, 0] := Pattern[CharPos];
Token[NumTokens, 1] := OC;
Ptr := CharPos+1;

ELSE
END; (* CASE *)
INC(CharPos)
END; (* WHILE *)
(* Store any trailing characters *)
IF
CharPos > Ptr THEN

    INC(NumTokens);
    Copy(Pattern, Ptr, (PatternLength-Ptr+1), Token[NumTokens]);
END;
END ScanPattern; (* Scan_Pattern *)

PROCEDURE LocatePattern(): INTEGER;
VAR
I, FirstChar, Ptr, Ptr2: INTEGER;
VAR LocatePatternResult: INTEGER;
BEGIN
FirstChar := 0;
(*--> Ptr := 1; *) Ptr := 0;
(*--> Ptr2 := 1; *) Ptr2 := 0;
I := 1;
WHILE I <= NumTokens DO

    IF
        (*--> INTEGER(Pos('?', Token[I])) > 0 THEN *)
        Pos('?', Token[I]) <= HIGH(Token[I]) THEN

            (* Sub-pattern has one or more '?' *)
            INC(Ptr, INTEGER(Length(Token[I])));
            (* does the text following the '?..?' match ? *)
            Ptr2 := OffsetPos(Line, Token[I+1], Ptr);
            IF Ptr <> Ptr2 THEN
                (*--> Ptr2 := 0 *) Ptr2 := HI
            END;
            INC(I);
        END;
        (*--> IF (INTEGER(Pos('?', Token[I])) > 0) OR (Token[I] <> '**') THEN *)
        IF (Pos('?', Token[I]) <= HIGH(Token[I])) OR
            (Token[I, 0] <> '*') THEN
            Ptr2 := OffsetPos(Line, Token[I], Ptr)
        END;
        (*--> IF (Token[I] <> '**') THEN *)
        IF Token[I, 0] <> '*' THEN

            IF (*--> (Ptr2 = 0) AND (FirstChar > 0) THEN *)
                (Ptr2 = HI) AND (FirstChar < HI) THEN
                    (*--> FirstChar := 0; *) FirstChar := HI;

            DEC(I, 1);

            ELSIF (*--> (Ptr2 = 0) AND (FirstChar = 0) THEN *)
                (Ptr2 = HI) AND (FirstChar = HI) THEN
                    I := NumTokens
            ELSE
                IF (*--> (FirstChar = 0) *)
                    (FirstChar = HI) THEN
                        FirstChar := Ptr2
                END;
                Ptr := Ptr2+INTEGER(Length(Token[I]));
            END;
            END;
            INC(I);
        END; (* WHILE I *)
        LocatePatternResult := FirstChar;
        RETURN LocatePatternResult
END LocatePattern; (* Locate_Pattern *)

VAR PatternSearchResult: INTEGER;
BEGIN (* Pattern_Search *)
ScanPattern;

```

(continued on page 84)

A POWERFUL UTILITY

Programmer's Journal is more than a "computer magazine." It is also a powerful utility for the serious PC programmer.

- PJ is valuable subroutines & efficient source code that you can use immediately.
- PJ is high quality program listings in ASM, C, Pascal, BASIC, dBase® and more that can save you time and money.
- PJ is handy information on DOS, graphics, memory resident programs and more.
- PJ is hot technical tips from experts like Tom Swan, Michael Abrash, William Hunt, and Rex Jaeschke.
- PJ is helpful guidance for consultants from Paul Barkley, useful industry news from Frank Greco and from Hal (*DTACK GROUNDED*) Hardenbergh.

RESPECTED BY PROFESSIONALS

"For hot programming tips look to Doctor Dobbs Journal or the excellent but relatively little known *Programmer's Journal*."

Peter Norton —*Inside the IBM PC*

"...stuffed with useful information, it definitely deserves a look."

Ray Duncan—*DDJ* 6-86

"I really love this magazine!"

Harry Miller —Editor, *PC WORLD*

GET ONE FREE!

To see what *PJ* has to offer just do one of the following:

- call us at (503) 484-2162
- circle our Reader Service No.
- return the coupon at right

We'll send you a **FREE** sample copy of our latest issue, reserve a 1 year subscription (5 more issues), and invoice you for \$19.95. If *PJ* is not the utility you need, simply write "CANCEL" on the bill, return it. Keep your free issue and owe nothing.

Programmer's Journal
29 W. 29th Suite 5
Eugene, OR 97405

Please allow 4-6 weeks for delivery of first issue. Offer good in U.S. only. Foreign subscriptions must be prepaid in U.S. funds. Can/Mex \$29.95, elsewhere \$39.95.

Circle no. 367 on reader service card.

The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

JANUARY/FEBRUARY 1987 VOLUME 5.1 \$3.95

ADOS COMETH

The New DOS—Programming in the Protected Mode by Frank D. Greco

Michael Abrash—Inside The EGA

Tom Swan—How Many Holes Does It Take To Fill An IBM PC?

Bernard Robinson—Tips on Formatting BASIC Data

YES—Please send me a FREE sample issue of **PJ** and start my NO RISK subscription.

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

K4611

New
Release!

SEIDL MAKE UTILITY

Version 2.0

The BEST just got BETTER!

If you're serious about software development, consider the **SEIDL MAKE UTILITY (SMK)**. SMK is not just another copy of the Unix Make. It was specifically designed to deliver features and performance not found in other makes.

✓ **Structured Language** to describe dependencies in a clear, concise and portable manner.

✓ **Rich Command Set** includes parameterized macros, variables, if-then-else, iteration, wild cards, exception cases, macro libraries, interactive statements, environment access, pattern matching and much more!

✓ **Intelligent Analysis** algorithm handles nested include files, library dependencies, and performs consistency tests to detect errors that other makes would blindly ignore.

✓ **Seidl Version Manager** compatibility lets you expand your system into the most comprehensive revision/version control system available.

"SMK is a very good Make indeed. Its major distinction is a truly simple Dependency Definition Language, easy to learn and easy to use... you'll probably bless SMK."

—Sextant, July '86

"SMK offers many unique features. [The error handling facility] is extremely useful if a large number of files must be recompiled."

—Computer Language, June '86

DOS \$99.95 \$3.50 p&h
Version Only Call for other
op systems.

Call Today
1-313-662-8086

Visa/MC/COD Accepted
Dealer Inquiries Invited

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

Circle no. 114 on reader service card.

STRUCTURED PROGRAMMING

Listing Four (Listing continued, text begins on page 34.)

```
PatternSearchResult := LocatePattern();
RETURN PatternSearchResult
END PatternSearch; (* Pattern_Search *)

BEGIN (*----- M A I N -----*)
  ClrScr;
  DEFAULTLINE := 'Namir Clement Shammas';
  WriteString(stdinout, 'Default string is : ', 0);
  WriteString(stdinout, DEFAULTLINE, 0);
  WriteLn(stdinout);
  WriteLn(stdinout);
  WriteString(stdinout, 'Enter string ', 0);
  ReadBuffer(on);
  ReadString(stdinout, Line);
  ReadLn(stdinout);
  ReadBuffer(off);
  WriteLn(stdinout);
  IF (*--> Line = '' THEN *)
    Line[0] = OC THEN
    (*--> Line := DEFAULTLINE *)
    Assign(DEFAULTLINE, Line)
  END;
  WriteString(stdinout, 'Enter search pattern string ', 0);
  ReadBuffer(on);
  ReadString(stdinout, Pattern);
  ReadLn(stdinout);
  ReadBuffer(off);
  WriteLn(stdinout);
  WriteString(stdinout, 'Matches at position ', 0);
  WriteInt(stdinout, (PatternSearch(Line, Pattern) + 1), 0);
  WriteLn(stdinout);
  WriteLn(stdinout);
END PatternSearchTest.
```

End Listing Four

Listing Five

Listing 5. Turbo Pascal program that uses sets for character-counting histograms.

PROGRAM Sets;

```
(* Program for the demonstration of translating sets *)
(* by Namir C. Shammas *)
```

```
TYPE CharSet = SET OF CHAR;
  STRING30 = STRING[30];
  LSTRING = STRING[255];

VAR DigitSet, UpperCaseSet, LowerCaseSet : CharSet;
  OK, C : CHAR;
  I, J, Count_Digits, Count_Upper,
  Count_Lower, Count_Others : INTEGER;
  Filename : STRING30;
  Line : LSTRING;
  InFile : TEXT;
```

```
PROCEDURE Display_Histogram(Row, Count : INTEGER);
```

```
BEGIN
  GOTOXY((11 + Count div 100), Row);
  WRITE('*');
```

```
END;
```

```
BEGIN
  REPEAT
```

```
  DigitSet := ['0'..'9'];
  UpperCaseSet := ['A'..'Z'];
  LowerCaseSet := ['a'..'z'];
```

```
  WRITE('Enter filename ');
  READLN(Filename); WRITELN;
```

```
  ClrScr;
  WRITELN('Digits      ');
  WRITELN('Uppercase ');
  WRITELN('Lowercase ');
  WRITELN('Others     ');
  Count_Digits := 0;
  Count_Upper := 0;
  Count_Lower := 0;
  Count_Others := 0;
  Assign(InFile, Filename);
```

A Reconfigurable Programmer's Editor With Source Code

ME is a high quality programmer's text editor written specifically for the IBM PC. It contains features only found in the more expensive programmer's text editors. These features include:

- Multiple Windows
- Column cut and paste
- Regular Expressions
- C-like Macro Language
- Keyboard Macros
- Reconfigurable Keyboard

New commands and features may be added to the editor by writing programs in its macro language. The language resembles C, and it's extremely easy to write and examine macros when you are not dealing with a parenthesized language like LISP. The macro languages comes with a rich set of primitives for handling strings and changing the editor environment.

End Listing Five

```
Reset (InFile);

WHILE NOT EOF(InFile) DO BEGIN
  READLN(InFile,Line);
  FOR I := 1 TO Length(Line) DO BEGIN
    C := Line[I];
    IF C IN DigitSet THEN
      Count_Digits := Count_Digits + 1
    ELSE IF C IN UpperCaseSet THEN
      Count_Upper := Count_Upper + 1
    ELSE IF C IN LowerCaseSet THEN
      Count_Lower := Count_Lower + 1
    ELSE
      Count_Others := Count_Others + 1;

  END;

  Display_Histogram(1,Count_Digits);
  Display_Histogram(2,Count_Upper);
  Display_Histogram(3,Count_Lower);
  Display_Histogram(4,Count_Others);

END;
Close(InFile);
GOTOXY(1,20); WRITE('Want to scan another file? (Y/N) ');
READLN(OK);
UNTIL NOT (OK IN ['Y','y']);
GOTOXY(1,20); ClrEol; WRITELN('End of program');
END.
```

Listing Six

Listing 6. Modula-2 program that uses sets for character-counting histograms.

```
MODULE Sets;

FROM Strings
  IMPORT Assign, Insert, Delete, Pos, Copy, Concat, Length, CompareStr;
FROM ScreenHandler
  IMPORT ClrEol, ClrScr, DelLine, InsLine, GotoXY, WhereX, WhereY,
  CrtInit, CrtExit, LowVideo, NormVideo, HighVideo, SetAttribute,
  GetAttribute, normalAtt, boldAtt, reverseAtt, underlineAtt,
  blinkAtt, boldUnderlineAtt, blinkUnderlineAtt, boldBlinkAtt,
  reverseBlinkAtt, boldUnderlineBlinkAtt;
FROM TfileIO
  IMPORT Append, AssignFile, Close, Erase, Flush, Rename,
  Reset, Rewrite, Truncate, Eof;
FROM TTextIO
  IMPORT ReadInt, ReadCard, ReadChar, ReadString, ReadLn, ReadBuffer,
  WriteInt, WriteCard, WriteChar, WriteString, WriteBool, WriteLn,
  Eoln, SeekEof, SeekEoln;
FROM TKernelIO
  IMPORT File, FileType, OptionMode,
  StatusProc, ReadProc, WriteProc, ErrorProc,
  stdinout, input, output, con, trm, kbd, lst, aux, usr,
  constPtr, conInPtr, auxInPtr, usrInPtr, conOutPtr, lstOutPtr,
  auxOutPtr, usrOutPtr, errorPtr, IOresult, KeyPressed, IOBuffer,
  IOCheck, DeviceCheck, CtrlC, InputFileBuffer, OutputFileBuffer;
(* The following two IMPORT statements are manually added *)
FROM LongSet
  IMPORT BuildSet, InSet, SetOfChar, MakeEmptySet, Include;
FROM SYSTEM IMPORT WORD;

(* Program for the demonstration of translating sets *)
(* by Namir C. Shammas *)

TYPE
  STRING30 = ARRAY [0..30-1] OF CHAR;
  LSTRING = ARRAY [0..255-1] OF CHAR;

VAR
  DigitSet, UpperCaseSet, LowerCaseSet,
  YesNo (*--> this one is added *) : SetOfChar;
  OK, C: CHAR;
  I, J, CountDigits, CountUpper, CountLower, CountOthers: INTEGER;
  Filename: STRING30;
  Line: LSTRING;
  InFile: File;

PROCEDURE DisplayHistogram(Row, Count: INTEGER);
BEGIN
```

(continued on next page)

The source code option lets you see how text editors are written, and gives you real-world applications of various data structures. You will also learn how to write interpreters by examining code to the macro language compiler and interpreter.

A unique advantage of the source code option is that many programmers will have the opportunity to examine the code, improve parts of it, and add additional features. The best of these improvements and enhancements will be incorporated into future releases, which, as a registered source code user, you will receive updates for.

The code is written in C, and conforms to the ANSI standard. This allows the code to be compiled with any compiler that supports the ANSI standard. A few of the string handling and input-output routines are written in 8086 assembly language for speed.

The source code option is perfect for OEMs and VARs who want to add a text editor to their applications.

Price for program and on-line documentation —

\$34.95

Price for editor with complete source code —

\$84.95

Special offer — New York Word word processor —

\$19.95 — Split screen, mail merge, hyphenation, math, regular expressions

MAGMA
SYSTEMS

138-23 Hoover Ave., Jamaica, NY 11435

Circle no. 313 on reader service card.



SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation

Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

\$395.00

C Interpreter. Run the interpreter on any hardware and on any operating system. Develops true intermediate code, allowing full C features in an interpreter. User configurable interface to compiler library allows linkage with compiled routines.

HARDWARE AND FILE SYSTEM INDEPENDENT

KURTZBERG COMPUTER SYSTEMS

**41-19 BELL BLVD.
BAYSIDE, N.Y. 11361**

VISA/Master Charge accepted
(718) 229-4540

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp.
MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL Logo are trademarks of Kurtzberg Computer Systems

Circle no. 294 on reader service card.

STRUCTURED PROGRAMMING

Listing Six (Listing continued, text begins on page 124.)

```

GotoXY((11+(Count DIV 100)), Row);
WriteChar(stdinout, '*', 0);
END DisplayHistogram;

BEGIN
REPEAT
(* The following three statements were edited from the original lines *)
BuildSet(DigitSet,ORD('0'),ORD('9'));
BuildSet(UpperCaseSet,ORD('A'),ORD('Z'));
BuildSet(LowerCaseSet,ORD('a'),ORD('z'));

(* The next three lines are inserted to support sets for Yes/No answers *)
MakeEmptySet(YesNo);
Include(YesNo, WORD(ORD('Y')));
Include(YesNo, WORD(ORD('N')));

WriteString(stdinout, 'Enter filename ', 0);
ReadBuffer(on);
ReadString(stdinout, Filename);
ReadLn(stdinout);
ReadBuffer(off);
WriteLn(stdinout);
ClrScr;
WriteString(stdinout, 'Digits      ', 0);
WriteLn(stdinout);
WriteString(stdinout, 'Uppercase ', 0);
WriteLn(stdinout);
WriteString(stdinout, 'Lowercase ', 0);
WriteLn(stdinout);
WriteString(stdinout, 'Others      ', 0);
WriteLn(stdinout);
CountDigits := 0;
CountUpper := 0;
CountLower := 0;
CountOthers := 0;
AssignFile(InFile, Filename, text);
Reset(InFile, 0);
WHILE NOT Eof(InFile) DO

  ReadString(InFile, Line);
  ReadIn(InFile);
(* Loop limits were shifted by 1 from Pascal version *)
FOR I := 0 TO Length(Line)-1 DO

  C := Line[I];
(* InSet() is used to test char C instead *)
(* of BITSET{} generated by the Translator *)
  IF InSet(DigitSet,ORD(C)) THEN

    INC(CountDigits, 1)
  ELSIF InSet(UpperCaseSet,ORD(C)) THEN

    INC(CountUpper, 1)
  ELSIF InSet(LowerCaseSet,ORD(C)) THEN

    INC(CountLower, 1)
  ELSE
    INC(CountOthers, 1)
  END;
END;
DisplayHistogram(1, CountDigits);
DisplayHistogram(2, CountUpper);
DisplayHistogram(3, CountLower);
DisplayHistogram(4, CountOthers);
END;
Close(InFile);
GotoXY(1, 20);
WriteString(stdinout, 'Want to scan another file? (Y/N) ', 0);
ReadBuffer(on);
ReadChar(stdinout, OK);
ReadLn(stdinout);
ReadBuffer(off);
UNTIL NOT (InSet(YesNo, ORD(OK))); (* Boolean expression was edited *)
GotoXY(1, 20);
ClrEol;
WriteString(stdinout, 'End of program', 0);
WriteLn(stdinout);
END Sets.

```

End Listing Six

Listing Seven

Listing 7. Turbo Pascal program that performs direct screen memory access by using simple absolute variables.

```

Program Screen;

(* Program to demonstrate direct memory access in Turbo Pascal *)

TYPE STRING80 = STRING[80];

VAR Message : STRING80;
    Row, Col : INTEGER;

PROCEDURE DISP_STR(S : STRING80; Row, Col : INTEGER);
(* Procedure to write a string to the screen memory *)

TYPE SCREEN80 = ARRAY [1..25,1..80,1..2] OF CHAR;

VAR DISP : SCREEN80 Absolute $B000:0000;
    (* For color display use *)
    (* DISP : SCREEN80 Absolute $B800:0000; *)
    I, J : INTEGER;

BEGIN
    J := Length(S);
    FOR I := 1 TO J DO
        DISP[Row,Col + I - 1,1] := S[I];
END;

BEGIN
    ClrScr;
    WRITELN('Enter message ');
    READLN(Message);

    ClrScr;
    Col := 1;
    FOR Row := 22 DOWNTO 1 DO
        DISP_STR(Message,Row, Col+Row);
    WHILE NOT Keypressed DO;
        ClrScr;
    END.

```

End Listing Seven

Listing Eight

Listing 8. Modula-2 program that performs direct screen memory access by using simple absolute variables.

```

MODULE Screen;

FROM Strings
    IMPORT Assign, Insert, Delete, Pos, Copy, Concat, Length, CompareStr;

FROM ScreenHandler
    IMPORT ClrEol, ClrScr, DelLine, InsLine, GotoXY, WhereX, WhereY,
    CrtInit, CrtExit, LowVideo, NormVideo, HighVideo, SetAttribute,
    GetAttribute, normalAtt, boldAtt, reverseAtt, underlineAtt,
    blinkAtt, boldUnderlineAtt, blinkUnderlineAtt, boldBlinkAtt,
    reverseBlinkAtt, boldUnderlineBlinkAtt;

FROM TTextIO
    IMPORT ReadInt, ReadCard, ReadChar, ReadString, ReadLn, ReadBuffer,
    WriteInt, WriteCard, WriteChar, WriteString, WriteBool, WriteLn,
    Eoln, SeekEof, SeekEoln;

FROM TKernelIO
    IMPORT File, FileType, OptionMode,
    StatusProc, ReadProc, WriteProc, ErrorProc,
    stdIn, input, output, con, trm, kbd, 1st, aux, usr,
    conStPtr, conInPtr, auxInPtr, usrInPtr, conOutPtr, 1stOutPtr,
    auxOutPtr, usrOutPtr, errorPtr, IOresult, KeyPressed, IOBuffer,
    IOCheck, DeviceCheck, CtrlC, InputFileBuffer, OutputFileBuffer;

(* Program to demonstrate direct memory access in Turbo Pascal *)

TYPE
    STRING80 = ARRAY [0..80-1] OF CHAR;

```

(continued on next page)

#1 Lint for MS-DOS

**KILLS
C BUGS
FAST**

PC-lint

**The professional
diagnostic facility for C**

PC-lint lets you zap swarms of C bugs and glitches at a time.

Now you can uncover the quirks, inconsistencies, and subtle errors that infest your C programs . . . waiting to bite you. PC-lint finds them all . . . or as many as you want . . . in one pass. Set PC-lint to match your own style.

Outperforms any lint at any price

- Full K&R support and common ANSI enhancements (even MS keywords)
- Finds inconsistencies (especially in function calls across multiple modules!)
- Modifiable library descriptions for 8 popular compilers
- Super fast, one-pass operation
- Suppress any error message
- Millions of options

PRICE \$139 • MC • VISA • COD

Includes USA shipping and handling.
Outside USA, add \$15. In PA add 6%.

**ORDER TODAY,
30-day guarantee**

Runs under MS-DOS 2.0 and up, and AmigaDOS. Uses all available memory.

Trademarks: PC-lint (Gimpel Software).
MS, MS-DOS (Microsoft). Amiga (Commodore).

GIMPEL SOFTWARE

3207 Hogarth Lane,
Collegeville, PA 19426
(215) 584-4261

STRUCTURED PROGRAMMING

Listing Eight (Listing continued, text begins on page 124.)

```
VAR
  Message: STRING80;
  Row, Col: INTEGER;

(* Procedure to write a string to the screen memory *)
PROCEDURE DISPSTR(S: ARRAY OF CHAR; (* Edited *)
  Row, Col: INTEGER);

TYPE
  SCREEN80 = ARRAY [1..25], [1..80], [1..2] OF CHAR;

VAR
  DISP [0B000H:00H] : SCREEN80;
  (* For color display use *)
  (* DISP[0B800H:00H] : SCRENN80 *)

  I, J: INTEGER;

BEGIN
  J := Length(S);
  (*--> FOR I := 1 TO J DO *)
  FOR I := 0 TO J-1 DO
    DISP[Row, Col+I, 1] := S[I] (* Edited *)
  END;
END DISPSTR;

BEGIN
  ClrScr;
  WriteString(stdinout, 'Enter message ', 0);
  WriteLn(stdinout);
  ReadBuffer(on);
  ReadString(stdinout, Message);
  ReadLn(stdinout);
  ReadBuffer(off);
  ClrScr;
  Col := 1;
  FOR Row := 22 TO 1 BY -1 DO
    DISPSTR(Message, Row, Col+Row)
  END;
  WHILE NOT KeyPressed() DO
  END;
  ClrScr;
END Screen.
```

Listing Nine

Listing 9. Modula-2 program that displays and alters the states of the [Caps Lock] and [Num Lock] keys on an IBM PC or compatible.

Here's why you should choose Periscope as your debugger...

You'll get your programs running fast. "It works great! A problem we had for three weeks was solved in three hours," writes Wade Clark of MPPi, Ltd.

You'll make your programs solid. David Nanian says, "I can't live without it!! BRIEF, a text editor my company wrote, would not be as stable as it is today without Periscope."

You'll protect your investment. We won't forget you after the sale. You'll get regular software updates, including a FREE first update and notice of later updates. You'll get technical help from Periscope's author. And you'll be able to upgrade to more powerful models of Periscope if you need to. One Periscope user writes, "...

your support has won over even the heart of this hardened programmer!"

You deserve the best. Thousands of programmers rely on the only debugger that PC Tech Journal has ever selected as **Product of the Month** (1/86). You owe it to yourself to find out why, first hand.

You can try it at no risk. You get an unconditional 30-Day, Money-Back Guarantee, so you can't lose.

Start saving time and money now — order toll-free, 800/722-7006. Use MasterCard, Visa, COD, or a qualified company purchase order. As one user puts it, Periscope is "one of the rare products, worth every penny!"

Periscope I, software, manual,
protected memory board and
breakout switch \$295

Periscope II, software, manual, and
breakout switch \$145

Periscope II-X,
software and manual \$115

Add shipping - \$3 US; \$8 Canada; \$24 elsewhere.
Ask about air shipment if you can't wait to get
your programs up and running!

PERISCOPE

The Periscope Company, Inc.

(formerly Data Base Decisions)

14 Bonnie Lane, Atlanta, GA 30328 404/256-3860

MULTITASKING

Introducing

MultiDos Plus

The new multitasking software
for the IBM-PC.

Ideal for developing applications
in process control, data acquisition,
communications, and other
areas. Check these features which
make **MultiDos Plus** an unbeatable
value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in any language.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
 - * Intertask message communication. Send/receive/check message present on 64 message queues.
 - * Task control by means of semaphores. Get/release/check semaphores.
 - * Change priority-128 priority levels.
 - * Suspend task for specified interval.
 - * Spawn and terminate external and internal tasks.
 - * Disable/enable multitasking.
 - * and more!
- Runs most commodity software including: Lotus 1-2-3, Dbase, Wordstar, and others.
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Monochrome/CGA display adaptors or equivalent cards only. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

ONLY \$29.95 +\$2.95 S/H

Outside USA add \$7.95 shipping and handling.
Visa and Mastercard orders call toll-free: 1-800-367-6707. In Mass call 617-651-0091, or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760

MA orders add 5% sales tax. Write for source code and quantity price.

Circle no. 309 on reader service card.

```
MODULE KeyLock;

(*
 * This program interactively displays and changes the states
 * of the Caps Lock and Num Lock keys on an IBM PC or compatible.
 * It is particularly useful with keyboards that have LEDs on these
 * keys; these sometimes get out of sync.

*
 * This program is adapted from FL.COM by Patrick Swayne, from
 * the June 1986 issue of REMark magazine.
 *
 * The declaration of keyflag is a feature of the Logitech compiler.
 *

 * James Janney, June 1986
 *)

FROM Terminal IMPORT ReadString, WriteString, WriteLn;

CONST
    NumLock = 5;          (* Num Lock bit *)
    CapsLock = 6;         (* Caps Lock bit *)

VAR
    keyflag [40H:17H] : BITSET; (* ROM-BIOS keyboard status word *)
    cmd : ARRAY [1..80] OF CHAR;

BEGIN
    LOOP
        WriteString("1. Caps Lock is ");
        IF CapsLock in keyflag THEN
            WriteString("on")
        ELSE
            WriteString("off")
        END;
        WriteLn;

        WriteString("1. Num Lock is ");
        IF NumLock in keyflag THEN
            WriteString("on")
        ELSE
            WriteString("off")
        END;
        WriteLn;

        WriteString("Enter option to change: ")
        ReadString(cmd);

        CASE cmd[1] OF
            '1' : keyflag := keyflag / {CapsLock}      (* toggle Caps Lock *)
            | '2' : keyflag := keyflag / {NumLock}       (* toggle Num Lock *)
        ELSE
            EXIT
        END;

        WriteLn; WriteLn;
    END;
END KeyLock.
```

End Listings

Nroff: Hashing, Expressions, and Roman Numerals

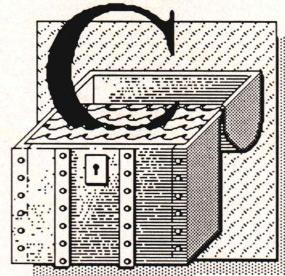
After the shell was published in this column, I vowed that I'd stay away from long programs. Publishing them was just too much work. I'm going to break my own rule for the next few months, however, by presenting nr, my version of the Unix text formatter, nroff. There are two reasons for this. First, nr includes a bunch of subroutines that are useful even if you're never going to use it as a word processor. Good examples are the hash table management functions and the general-purpose expression analyzer, which I'll discuss this month. Second, the program is an almost complete implementation of nroff and it includes several of troff's features as well.¹ It does hyphenation and proportional spacing, it can format equations and matrices, and it's easily configurable to most printers. I have it driving a Brother HR-15 (a Diablo-compatible daisy-wheel printer), a Thinkjet, and an HP Laserjet+ laser printer. Nr can do as much as many word processors that cost several hundred dollars. It is also considerably more powerful than any of the various roff spin-offs available from various users' groups and bulletin boards. Unlike most of these, it doesn't derive from the word processor in *Software Tools in Pascal*.²

I hadn't published the program before now because I'd just about con-

by Allen Holub

vinced myself that it was obsolete. My research into the newer wizzywig editors has convinced me otherwise, however.

Text formatters, like many classes of computer tools, can be divided into two categories. The *interpretive* formatters (such as WordStar or Microsoft Word) combine text entry or edit-



ing together with text formatting: adjusting margins and line length, hyphenating, and so on. The *compiled* word processors, such as nroff and TeX work like compiled languages do. You create an ASCII input file composed of mixed text and formatting commands and then submit this file to a second program that actually does the formatting.

Both these techniques have their advantages and disadvantages—something that was made painfully clear to me a few months ago when I tried to typeset a book with Version 3.0 of Microsoft Word. I really like the idea of being able to see what the document will look like as I'm typing it. Word, for example, actually displays italicized text in italics on the screen. Using Word turned out to be a serious mistake, however. It may do a nice job on a newsletter or a medium-length paper, but when it comes to something real, such as a book, it just can't do the job. If you can't type, if you're afraid of computers, and if you're never going to try to create a document longer than a newsletter, Word is the program for you. If any of the above don't apply to you, stay away from it.

First of all, it takes two to three times longer for me to enter text using Word than it does with my normal editor. I'm a touch-typist—if I don't have to lift my hands from the home row of the keyboard, I can type about 85 words a minute. Microsoft Word doesn't permit this, however.

Many things have to be done with the function keys. Then there's that idiot mouse. The nonmouse commands are so involved that they're almost useless. Many of them use ten or more keystrokes to do a simple action such as deleting a block of text. The mouse is really an integral part of the program. So, every time you need to delete some text, change a font, or whatever, you have to let go of the keyboard, find the idiot mouse, which has buried itself under piles of paper, clear off an area of desk big enough for the thing to move, and then execute the command. If you can't type, this might not be a problem, but I'd think that the mouse-related problems alone would severely limit the utility of Word in a normal office environment. All this is compounded by Word's refusal to implement even the simplest editing control codes. For example, Word does not recognize Ctrl-H as a backspace—you have to find the backspace key, which is way up in the boondocks on the IBM keyboard. I found I couldn't type anything without Pro-Key installed under Word, and even then text entry was difficult.

The next problem is that Word doesn't really show you what you're getting. It doesn't display text in the correct point sizes. It lets you see how many words will be on the output line, but when you do this, the tab stops don't display correctly and the right margin isn't adjusted anymore. Word doesn't really format the text as you type either—at least, it reformats the text as part of the printing process. The "compute page break" command takes as long to execute as my compiled word processor takes to format the entire document.

The final problem, and the one that made the program utterly

worthless to me, is that Word evidently keeps the entire document in memory at once. Though I have a 640K system, this isn't enough to typeset a book. The program just can't handle that much text. Word has a text-merge feature, but evidently it can't be used if you're assembling an index or table of contents.

So I filed Word in the circular file and went back to my own text formatter, nr. It supports all the things you need to do most word processing applications—multiple fonts and point sizes, multicolumn text, hyphenation, proportional spacing, both footnotes and endnotes, automatic index and table of contents generation, and so forth. I can use my normal editor to enter text.³ Moreover, nr supports an extensive macro language that effectively lets you change the way the program works to suit your needs of the moment. Most nr (and nroff) input documents don't actually use the primitive commands supported by the word processor itself; rather, they use various macro packages that use these primitives. The macros can be used in the same way as you use subroutines; you can pass them arguments, define and modify global variables, and so forth. Though WYSIWYG isn't available, nr supports an adequate screen-preview mode. I've configured it so that, when ANSI.SYS is installed, underlined text is displayed on the screen underlined, boldface is shown in high-intensity video, and overstruck text blinks.

It's going to take several months to get the whole program printed in this column, so you may find include file or subroutine references that aren't part of the current month's listings. I'll try to minimize this, but a little forward referencing is inevitable. The complete program will be available on a disk from *DDJ* in March. The remainder of this month's column is a discussion of several nr support routines. In the next column I'll give a detailed users' manual along with more code. I'll finish the code in a third column and also present a detailed program description at that time (once all the code is available).

Much of the support code for nr has already been published in previous columns. This puts me in some-

thing of a quandary because most of this code has undergone minor modifications since its original publication. After a great deal of soul-searching, I've decided not to reprint these routines, even though there are minor changes. The algorithms are basically the same, so the earlier articles will serve as an adequate reference. I'm also assuming that no one in their right mind would actually type the

Included in nr are subroutines that are useful even if you never use it as a word processor.

whole thing in by hand when the entire program will be made available on a disk. Nr uses the following, previously published, support routines:

DDJ, May 1985—stoi(), getargs()
DDJ, June 1985, queues—makequeue(), show_next(), enqueue(), dequeue(), sp_used()
DDJ, June 1985, bit maps—setbit(), testbit(), makebitmap()
DDJ, October 1985, hyphenation—hyphen()

Most of the support routines printed this month are adequately commented and can stand alone. There are three exceptions: A set of general-purpose, symbol table maintenance functions; a binary-to-ASCII conversion function that can print in Roman numerals and English words as well as the usual digits; and a powerful expression analyzer.

Symbol Table Maintenance with Hashing

Listing Three, page 52, presents a set of symbol table maintenance functions that use a hashing strategy for table maintenance.⁴ They were originally written for a compiler project

but are useful anywhere you need a list of objects ordered by name. I've made the routines as general purpose as possible. As was the case in the AVL tree routines printed in the August 1986 issue of *DDJ*, I've separated the actual maintenance functions from the part of the program that's going to use the table. That is, all knowledge of how the table is organized is hidden within the maintenance functions. The application program doesn't know (or care about) the actual details. By the same token, the maintenance functions don't know about anything the application is doing with the table. All they know about is how to put things in tables and take them out again.

All the routines use the hash.h file given in Listing Four, page 58. For now let's avoid the details and consider the one thing that's needed from this file to use the maintenance functions—the HASH_TAB structure. A HASH_TAB is used in the same way as the FILE structure used by the buffered I/O routines is used. The subroutine:

```
HASH_TAB *maketab( size )
unsigned size;
```

is analogous to *fopen()*; it creates a table having the indicated *size* and returns a pointer to it. It turns out that hash functions work best if *size* is a prime number. Some likely candidates are 47, 61, 89, 113, 127, 157, 193, 211, 257, 293, 359, and 401. The larger the table, the quicker the access time. On the other hand, the larger the table, the larger the table. I've found 127 and 257 to be reasonable numbers. *Maketab()* won't return if it can't make the table (it exits with a status of 1 in this case).

Once a table has been created, objects can be inserted with a call to:

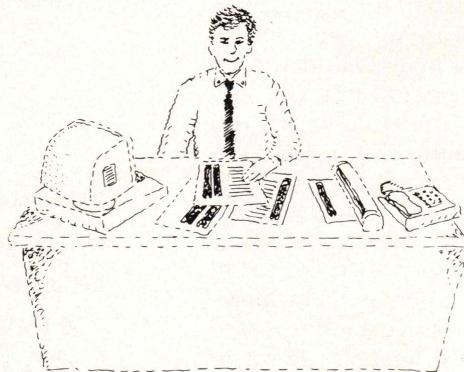
```
char *addsym( tabp, name, size )
HASH_TAB *tabp;
char *name;
```

where *tabp* is a pointer returned from a previous *maketab()* call, *name* is a symbol name, and *size* is the size of a block of memory that the application will use. Only the first 32 characters in *name* are significant; additional characters are ignored. *Addsym()* won't return if it can't get

SOFTSTRIP® NOW OFFERS YOU SOMETHING NEVER BEFORE AVAILABLE...



CONVENTIONAL DATA HANDLING



THE SOFTSTRIP SYSTEM

A CHOICE.

Until now you were stuck with disks.

No more. Install our unique STRIPPER™ software on your personal computer today and discover the many benefits of the fastest, easiest, least expensive way to handle information.

STRIPPER lets you print—ON PAPER—your own machine readable Softstrip data strips using your dot matrix printer. The Softstrip System Reader reads that information into a computer rapidly. With STRIPPER and the reader, your PC and printer become part of the most versatile information handling system available.

With this system you can do anything you wish with any data you have in your PC—ON PAPER.

DATA ENTRY: Why use keystrokes when you can eliminate them with data strips? Whatever the document - invoices, packing slips, memos, letters, sales reports, the list is endless—simply print a data strip right on the same printed page. Now you have a document that is both human readable and machine readable. A typical document can be entered in only 15 seconds using data strips. And, it ends keystroke errors forever.

DATA DISTRIBUTION: Why copy disks? It's time consuming and expensive. Softstrip data strips will end all that. Simply photocopy as many data strips as you like and send them by mail. Data strips ignore folding, coffee stains, ink marks and, by the way,

magnetic fields. And if you're using telecommunications, you can stop making the phone company rich.

DATA STORAGE AND RETRIEVAL: Why have a file of disks and a file of paper? Eliminate one with Softstrip data strips. File the data strip with the document. Better still, print the strip right on the document. Then put it in a file or binder.

Retrieval is simple. To find existing data, pull the document and its related data strip from the file. They've been stored together. Then use the reader to enter the data. No more hassle trying to match documents with the right disk—if you can find it.

DATA TRANSFER: Why bother with cables, modems and phone lines to move files between computers? A Softstrip data strip generated by an IBM PC can be read into another PC, or compatible, an Apple or even a Macintosh. If you work at home on a Macintosh, make a data strip on your printer, take it into the office and read it into your IBM PC. Simple. And we've created the utilities to let you do that easily. (See Application Notes on opposite page.)

Fascinating, isn't it? Anything you can do with disks can be done with the Softstrip data strip system—faster, easier and at lower cost—ON PAPER.

All you need is STRIPPER software at \$19.95 and the Softstrip System Reader at \$199.95.



The
PC
User's
Edge

NOW! TRANSFER DATA – PROGRAM TO PROGRAM WITH **SOFTSTRIP®**

Now you can move data between programs quickly and easily using SOFTSTRIP data strips.

Using the Softstrip System, you can move data between computers and such programs as WordStar and MacWrite, dBASE and AppleWorks, Lotus 1-2-3 and Excel and ReadySetGo and many others.

We've created a series of several dozen Application Notes on Softstrip data strips. These lead you through simple steps to make the file transfer as easy as possible, adding even more versatility to your personal computer when you purchase the SOFTSTRIP SYSTEM. The advanced system you've been hearing so much about.

All you need to move data between programs is STRIPPER™ software at **\$19.95** and the Softstrip System Reader at **\$199.95**.

For a complete list of Application Notes, contact your dealer or call Cauzin.

FEBRUARY CASE HISTORY

Elroy Bond, Vice President and Director of Data Processing at Golden State Mutual Life Insurance had a problem with data transmission. The difficulty was his telecommunications equipment. No matter how reliable and easy-to-use his PC based system was, if the phone lines were down or his PC operator was ill, he would end up sending disks through the mail from their branch offices in Winston-Salem, N.C. and Compton, Westwood, and Los Angeles, CA. Besides the hassle of copying a disk and using a disk mailer, there was always the possibility of having to resend the disk because of damaged data.

Now the branch offices each have their own copy of the STRIPPER™ printing program. When their modem based system fails to operate, they just send a document in data strip format. Bond is planning on installing SOFTSTRIP Readers at every location, so that when he updates their software, all he has to do is print out a data strip document on his Epson printer, photocopy it and mail it in an envelope for 22 cents.

Users' Groups: Call for Special User Group Discounts.

ACT NOW! Don't delay. See your local Softstrip dealer or call us at 1-800-533-7323. In Connecticut: 203-573-0150.

CAUZIN

835 South Main Street
Waterbury, CT 06706
(203) 573-0150

For Europe and Asia Contact:

Softstrip International, Ltd.
53 Bedford Square
London, WC1 B3DP England
01-631-3775 Telex: 263874SOFTST G

This data strip contains IBM2MAC, a utility that runs on the IBM and converts an IBM file to Macintosh format.

(continued from page 91)

the memory. It prints an error message and exits from the program with a status of 1 in this case. On success, *addsym()* returns a pointer to a block of memory that can be used by the application just as if it had been returned from *malloc()*. The symbol name is stored automatically by *addsym()*, so it need not be stored again in the application area. Note that the memory is already attached to the table at this point. That is, the allocate and add functions have been merged into a single subroutine.

The application can look for a specific symbol with a call to:

```
char      *findsym( tabp, name )
HASH_TAB *tabp;
char      *name;
```

where *tabp* is a pointer returned from *maketab()* and *name* is a symbol name. If the named symbol is in the table, the same pointer that was returned from the original *addsym()* call is returned; otherwise, *NULL* is returned. The application can use this pointer as it sees fit. Note that *findsym()* does not remove the node from the table—it just returns a pointer to the node's application area.

A symbol is deleted by calling:

```
delsym( tabp, symp )  
HASH_TAB *tabp;  
char *symp;
```

where *tabp* is, again, a pointer returned from *maketab()* and *symp* is a pointer returned from *findsym()*. *Delsym()* both removes the symbol from the table and deletes all memory associated with that symbol, including the application area.

The basis of all hash strategies is the conversion of a string into a random number. This number is then used as an index into a large array—the hash table itself. In the best hash functions, there is virtually no relation between the original string and the hashed number. The purpose of the ran-

domization is to scatter the names as evenly as possible throughout the entire table. In practice, there are always a few strings that will hash to the same numeric value. This condition, called a *collision*, can be resolved in several ways. The easiest is to make the hash table itself an array of pointers to structures. Each of these points to the head of a linked list. When a collision occurs, the new node is just linked into the head of the list.

This strategy is particularly convenient in a compiler's symbol table, where two symbols might have the same name but different scope. If colliding nodes are linked to the head of the list, they will shadow previously declared nodes associated with the same name.

The *HASH_TAB* and *BUCKET* structures defined in hash.h are used to define the table. A *HASH_TAB* is the actual table. It looks like:

```

typedef struct HASH_TAB
{
    BUCKET **table      ;
    int      size       ;
    int      numsyms   ;
}
HASH_TAB;

```

The *table* field points at the actual array. This array is created with a *malloc()* call when *maketab()* is called. *Size* holds the number of elements in the table, as was passed to *maketab()*. *Numsyms* is a count of the number of symbols currently in the table. It's useful only for statistical purposes—the hash functions themselves don't use it.

Each element of the hash table array points at a *BUCKET* structure, defined in hash.h as:

```
typedef struct element_
```

```

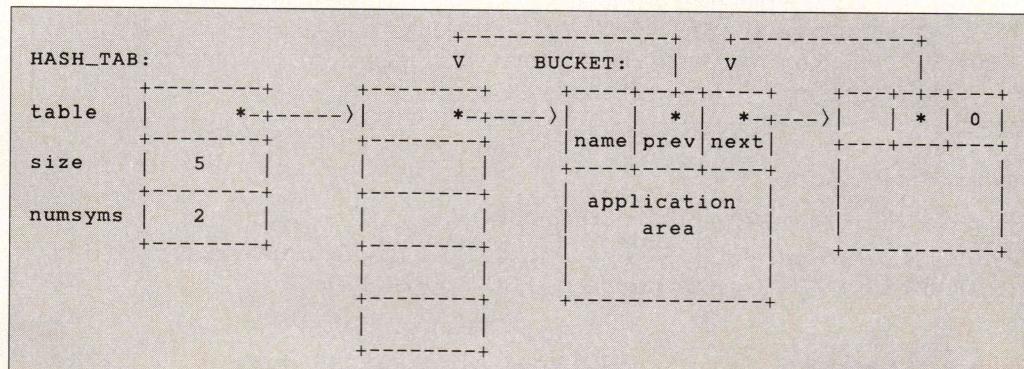
    {
        struct element_ *next;
        struct element_ **prev;
        char sname[ MAXNAME + 1];
    }

```

BUCKET:

A *BUCKET* is actually a header, pre-fixed onto the top of a block of memory used by the application program. This technique is used by *malloc()*, and I used the same technique in the AVL tree routines a few months ago. *Addsym()* returns a pointer to the application area, just below the *BUCKET*. The *BUCKETS* themselves hold the symbol name and two pointers used to form a doubly linked list.

The whole system of structures is shown in Code Example 1, below. The example shows a length 5 table with two symbols inserted into it. The symbols both hash to the same number, so a collision condition is shown. The hash table itself is an unnamed array pointed to by the *table* field of the *HASH_TAB* structure. The array entry forms the head of a linked list, pointed to by the *next* fields in the various *BUCKETS*. The list is double-linked, which means that not only is there a pointer to the next node in the chain but also a pointer to the previous node is kept. This lets you remove a symbol from the list without having to chase down the entire list. All you need is a pointer to the actual node you wish to delete. Note that the *prev* pointer points not at an entire *BUCKET* structure but rather at the *next* field of the previous structure. This way, the leftmost node in the chain is not a special case. All backward pointing pointers point at objects of the same type—pointers to *BUCKETS*. Assuming that *p* is a



Code Example 1: A complete hash table

C Programmers!

db_VISTA™: high-speed Database written exclusively for C NOW offers SQL-based Query

"db_VISTA™ has proved to be an all-round high performer in terms of fast execution..."

John Adelus, Hewlett-Packard Ltd./Office Productivity Division

High-speed data retrieval and access... just two benefits of using RAIMA's network model DBMS, db_VISTA. Combine these benefits with those of C-speed, portability, efficiency, and you begin to understand db_VISTA's real measure... performance.

db_QUERY™: new simplicity retains performance!

db_QUERY, our new C-linkable, SQL-based, ad-hoc query and report writing facility... provides a simple, relational view of db_VISTA's complex network database. No longer will you give up performance for simplicity... combine db_QUERY with db_VISTA... you have both!

Independent Benchmark proves High-Speed model 2.76 times faster

An independent developer benchmarked db_VISTA against a leading competitor. Eleven key retrieval tests were executed with sequentially and randomly created key files.

*Result of 11 Key Retrieval Tests

db_VISTA : 671.24 seconds
Leading Competitor : 1,856.43 seconds

db_VISTA's high-speed network database model lets you precisely define relationships to minimize redundant data. Only those functions necessary for operation are incorporated into the run-time program.

Application Portability Complete Source Code

For maximum application portability, every line of db_VISTA's code is written in C and complete source code is available. db_VISTA operates on most popular computers and operating systems. So whether you write applications for micros, minis, or mainframes... db_VISTA is for you.

How db_VISTA works...

Design your database and compile your schema file with the database definition language processor. Develop application programs, making calls to db_VISTA's C functions. Edit and review your database using the Interactive Database Access utility. Compile and link your C program with the db_VISTA run-time library, and your application is ready to run.

Multi-user and LAN capability

Information often needs to be shared. db_VISTA has multi-user capability and supports simultaneous users in either multi-tasking or local area networking environments, allowing the same C applications to run under UNIX, MS-DOS, and VAX VMS.



High-Speed Programming Tools,
Designed for Portability

"db_VISTA™ has proved to be an all-round high performer in terms of fast execution..."

Royalty-Free Run-Time

Whether you're developing applications for a few customers, or for thousands, the price of db_VISTA or db_QUERY is the same. If you are currently paying royalties for a competitor's database, consider switching to db_VISTA and say goodbye to royalties.

FREE Technical Support For 60 days

Raima's software includes free telephone support and software updates for 60 days. Technical support personnel are available to answer questions about our software or yours.

30-Day Money-Back Guarantee

Try db_VISTA for 30 days and if not fully satisfied, return it for a full refund.

Price Schedule

| | db_VISTA | db_QUERY |
|---------------------------|----------|----------|
| □ Single-user | \$ 195 | \$ 195 |
| □ Single-user w/Source | \$ 495 | \$ 495 |
| □ Multi-user | \$ 495 | \$ 495 |
| □ Multi-user w/Source | \$ 990 | \$ 990 |
| NEW: | | |
| □ VAX Multi-user | \$ 990 | \$ 990 |
| □ VAX Multi-user w/Source | \$ 1980 | \$ 1980 |

Call Toll-Free Today!

1 (800) db-RAIMA
(that's 1-800-327-2462)

---OR Call 1-206-828-4636



Read what others say...

"If you are looking for a sophisticated C programmer's database, db_VISTA is it. It lets you easily build complex databases with many interconnected record types. Raima's customer support and documentation is excellent. Source code availability and a royalty-free run-time is a big plus."

Dave Schmitt, President
Lattice, Inc.

"My team has developed a sophisticated PC-based electronic mail application for resale to HP customers. db_VISTA has proved to be an all-round high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us much time and development effort."

John Adelus, Hewlett-Packard Ltd.
Office Productivity Division

"On the whole, I have found db_VISTA easy to use, very fast with a key find, and powerful enough for any DBMS use I can imagine on a microcomputer."

Michael Wilson, Computer Language

db_VISTA Version 2.2

Database Record and File Sizes

- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- No limit on number of records or set types
- Maximum file size limited only by available disk storage
- Maximum of 255 index and data files

Keys and Sets

- Key length maximum 246 bytes
- No limit on maximum number of key fields per record—any or all fields may be keys with the option of making each key unique or duplicate
- No limit on maximum number of fields per record, sets per database, or sort fields per set
- No limit on maximum number of member record types per set

Operating System & Compiler Support

- **Operating systems:** MS-DOS, PC-DOS, UNIX, XENIX, SCO XENIX, UNOS, ULTRIX, VMS
- **C compilers:** Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, XENIX and UNIX

Features

- **Multi-user** support allows flexibility to run on local area networks
- **File structure** is based on the B-tree indexing method and the network database model
- **Run-time size**, variable—will run in as little as 64K, recommended RAM size is 256K
- **Transaction processing** assures multi-user database consistency
- **File locking** support provides read and write locks on shared databases
- **SQL-based** db_QUERY is linkable
- **File transfer** utilities included for ASCII, dBASE optional
- **Royalty-free** run-time distribution.
- **Source code** available.

Utilities

- Database definition language processor
- Interactive database access utility
- Database consistency check utility
- Database initialization utility
- Multi-user file locks clear utility
- Key file build utility
- Data field alignment check utility
- Database dictionary print utility
- Key file dump utility
- ASCII file import and export utility

*The benchmark procedure was adapted from "Benchmarking Database Systems: A Systematic Approach" by Bitton, DeWitt and Turbyfill, December 1983.

Call Toll-Free Today!
1 (800) db-RAIMA

3055-112th Avenue N.E. • Bellevue, WA 98004 USA • (206) 828-4636 Telex: 6503018237 MCI UW

Circle no. 206 on reader service card.

C CHEST

(continued from page 94)

pointer to a *BUCKET* that you want to delete, you can remove the *BUCKET* with:

```
if( *(p->prev) = p->next )
    p->next->prev = p->prev
```

The assignment in the *if* statement makes the *next* pointer of the previous node point around the current node. The next line is executed only if there's more than one node in the chain. In this case the *prev* field of the node that follows the one you want to delete is made to point at the *next* field of the node to the left of the one you want to delete. If the deleted node is at the head of the chain, the hash table array itself is modified.

One final point worth mentioning is the hash algorithm itself. More paper than I care to think about has been wasted talking about "efficient" hash algorithms. I've always been suspicious of these fancy functions, especially the ones that use several multiplies and divides. It doesn't seem that the inefficiencies inherent in the more complex algorithms can be countered by any better performance in the collision resolution department. How long can it take to chase down a linked list anyway? So being my own untrusting self, I tried out about 20 hash functions empirically. I was not surprised to find that the fanciest functions were so slow as to be virtually useless. The three best algorithms I tried are shown in Table 1, right. The first two are derived from the HashPJW function described in the "dragon" book.⁵ To my surprise, however, adding together the characters in the name is just about as good as HashPJW, and addition is a lot faster than all that shifting, type conversion, and XORing. I used addition in my own hash function. The keywords used in the test are names I extracted from various C programs in my own library.

Binary to Roman Numerals

Listing Five, page 60, shows *itoascii()*, a fancy binary-to-integer conversion routine. Unlike *itos()*, *ecvt()*, and the like, *itoascii()* can convert to several formats, depending on the value of its second argument. The various for-

mats are listed in Table 2, page 97.

The alphabetic output goes like:

a b c . . . y z aa ab ac . . . az ba bb
bc . . . bz ca . . .

It's useful if you're making outlines. The spelled out formats print:

one
two
three
. . .

thirteen

fourteen

fifteen

. . .

twenty-one

twenty-two

. . .

two thousand, one hundred thirty-one

two thousand, one hundred thirty-two

. . .

HASHPJW

```
for( h = 0; *name ; h = (h << 4) + *name++ )
    if( g = h & ((unsigned)(0) >> 4) )
        h = (h ^ (g >> WLEN-8)) ^ g;
```

1110 entries in 127 element hash table, 0 (0%) empty.

Mean chain length: 8, max=20, min=3, deviation=2

3 chains of length 3
7 chains of length 5
14 chains of length 7
15 chains of length 9
12 chains of length 11
5 chains of length 13
1 chains of length 15
1 chains of length 20

5 chains of length 4
11 chains of length 6
21 chains of length 8
22 chains of length 10
8 chains of length 12
1 chains of length 14
1 chains of length 17

SIMPLIFIED HASHPJW

```
for( h = 0; *name ; h = (h << 4) + *name++ )
    ;
```

1110 entries in 127 element hash table, 0 (0%) empty.

Mean chain length: 9, max=25, min=1, deviation=3

2 chains of length 1
4 chains of length 3
13 chains of length 5
12 chains of length 7
13 chains of length 9
9 chains of length 11
2 chains of length 13
1 chains of length 15
3 chains of length 17
1 chains of length 20
1 chains of length 25

4 chains of length 2
8 chains of length 4
5 chains of length 6
17 chains of length 8
16 chains of length 10
7 chains of length 12
4 chains of length 14
3 chains of length 16
1 chains of length 19
1 chains of length 21

ADDITION:

```
for( h = 0; *name ; h += *name++ )
    ;
```

1110 entries in 127 element hash table, 0 (0%) empty.

Mean chain length: 7, max=16, min=2, deviation=2

1 chains of length 2
1 chains of length 4
14 chains of length 6
16 chains of length 8
10 chains of length 10
7 chains of length 12
7 chains of length 14
1 chains of length 16

4 chains of length 3
9 chains of length 5
15 chains of length 7
21 chains of length 9
16 chains of length 11
4 chains of length 13
1 chains of length 15

Table 1: Performance of various hash functions

Roman numerals (i, ii, iii, iv, v, vi, vii, viii, ix, . . .) are somewhat limited. In particular, numbers larger than MMMMM can't be printed because an overscore would be required and most printers don't support overline, only underline. Also, because there's no zero in the Roman counting system, an Arabic 0 is used if n is zero.

The Arabic conversions are all done using *sprintf()*, spelled formats are done by *itoeng()* (line 35), *itoroman()* (line 128), and *itoalpha()* (line 202). Of these, *itoeng()* is the most straightforward. The main problems have to do with getting the commas and hyphens in the right place. The other two conversions are a little harder because neither number system has a zero in it. Consequently, you can't just modify a basic *itoa()* routine. *Itoranman()* uses a lookup table to assemble the numerals. (See Code Example 2, page 98.) The *Ms* used for thousands are printed with a simple loop on lines 178 to 181 of Listing Five. The rest of the digits are printed by the *for* loop, also in Code Example 2.

The outer loop executes at most three times. Rp starts out pointing to

the hundredths part of the table (the first ten entries). The line:

$$cp = *(rp + (n/i));$$

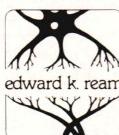
computes a pointer to the correct string, where n is the current value of the number and i is the current multiplier. It is 100 the first time through.

the loop, 10 the second time, and 1 the last time. The expression could also be written:

$$cp = rp[n/i];$$

N and i are then adjusted for the next iteration, and 10 is added to rp to make it point at the next part of the

Table 2: Itoascii() calling syntax



Transform Your Programs with **CPP—C Preprocessor Plus**

Includes ALL features of the standard C preprocessor.

- Define arbitrarily complex macros with #define command.
 - Include and nest files to any depth with #include command.
 - Include lines with #if, #ifdef and #ifndef commands.
 - Define multiple constants with enum command.
 - Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or * are ignored.)

Fast and flexible

- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
 - Can be used for any language, including assembler.
 - Can be used as a stand-alone macro/include processor.
 - Code can be used as the lexical analyzer for parsers or assemblers.

Complete[®]

- You get complete SOURCE CODE in standard C.
 - You get everything you need to use CPP immediately.
 - CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

Price: \$95.

Enteleki, Inc.
210 N. Bassett St., Room 101
Madison, WI 53703
Tele. (608) 258-7078

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5½ inch disk). Send a check or money order for \$95 (\$105 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, we DO NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check is included.

Circle no. 90 on reader service card

Megamax C

for the

Atari ST

"Don't even think about
another C compiler" Antic
Sept '89

"Don't even think about another C compiler" Antic Sept. '86

Mac-to-GS C & Mac-to-GS Pascal

Macintosh to Apple II GS cross compilers.
The fastest development systems for the II GS.

Megamax

Development Systems

Box 851521 • Richardson, TX 75085
(214) 987-4931 • Telex 5106018356

Circle no. 352 on reader service card

table. The number is printed by the inner *for* loop.

Itoalpha() is very *atoi()*-like. The lack of a zero complicates things here, too. In particular, the alphabetic numbers aren't a simple base-26 counting system. That is, if you map *a* to 0, *b* to 1, *c* to 2, and so forth, the output series would look like:

a b c d . . . ba bb bc . . . ca cb cc . . .

instead of:

a b c d . . . aa ab ac . . . ba bb bc . . .

Changing the mapping by leaving out the 0 and mapping *a* to 1, *b* to 2, and so forth gives you:

a b c d . . . a_ aa ab ac . . . b_ ba bb bc

which is closer to the desired series, but now the zero causes problems, represented by the underscore in the strings (placed where the zero would go if you had one). *Itoalpha()* actually does the conversion with:

do{

```
*p++ = (n % 26) +
(uppercase ? 'A' : 'a');
```

```
} while( (n = (n/26)-1) >= 0 );
```

Here *p* is a pointer to the output buffer. The *n % 26* selects the current digit in the expected way. You can't just divide by 26 to get the next digit because of that 0, so you compensate for the 0 by dividing and then subtracting 1 from the result. This subtraction

is done in the *while* part of the *do . . . while* statement.

Expression Analysis

An arithmetic expression analyzer, along with a discussion of the underlying theory, was published in the September 1985 C Chest. That analyzer had several problems that I knew about at the time but ignored in order to make the grammar more intuitively obvious. A real application, how-

```
static char *rnums[] =
{
    " ",      "C",      "CC",      "CCC",      "CD",
    "D",      "DC",      "DCC",     "DCC",      "CM",
    " ",      "X",       "XX",      "XXX",      "XL",
    "L",      "LX",      "LXX",     "LXXX",     "XC",
    " ",      "I",       "II",      "III",      "IV",
    "V",      "VI",      "VII",     "VIII",     "IX"
};

rp = rnums;
for( i = 10*10 ; n>0 && i>=1 ; i/=10 )
{
    cp = *(rp + (n/i));
    n %= i;
    rp += 10;
    for( ; *cp ; cp++)
        *dest++ = (uppercase) ? *cp : *cp + ('a' - 'A');
}
```

Code Example 2: Lookup table used by *itoroman()*

BRING YOUR HARD DISK BACK UP TO SPEED WITH H.D. TUNEUP!

The harder you work, the more files you put on your hard disk, the faster DOS works to fragment those files. Fragmented files make you wait longer for file loads and saves. **H.D. Tuneup** rearranges the data on your hard disk for the fastest possible file i/o times. Your disk will speed along like new.

"Much better than Disk Optimizer™"

- Philadelphia, PA

Requires IBM PC/XT/AT compatibility, DOS 2.x or higher and at least 196K. Hard disks up to 32mb may be tuned, along with most 5.25" diskettes.

ONLY \$39.95 + \$3.00 shipping (US/Canada)

SofCap Inc.

P.O. Box 131

Cedar Knolls, NJ 07927

Visa (201) 386-5876 MasterCard

If you want the absolute best possible performance from your hard disk:

TUNE IT UP WITH H.D. TUNEUP

Disk Optimizer is TM SoftLogic Solutions. *H.D. Tuneup* is TM SofCap.

DeSmet C execution profiler included

DeSmet C Compiler* still \$109

The professional's choice for fast compilation and execution. Includes Compiler, Assembler, Binder, Librarian, and Full Screen Editor (SEE™). Execution Profiler reports by address, procedure name or line number. Supports both disk and memory resident Overlays. Contains both 8087 and Software floating point support. Full STDIO library.

With D88 Debugger Option \$159

Gain most of the benefits of an interpreter while losing none of the run-time speed of the C88 compiler. Display C source and variable contents during execution. Set breakpoints by function name or line number. Examine and set variables by name using C expressions.

With Large Case Option and D88 ... \$209

Makes a great C Compiler even better. Adds 32-Bit Pointers to C88 so you can utilize all of your PC. Groups scalar and static data for fast access.

*D88 & Large Case Options available as add-ons.

C Ware Corporation

505 W. Olive, Suite 767, Sunnyvale, CA 94086 U.S.A.

(408) 720-9696 — Telex: 358185

We accept VISA, MasterCard & American Express

Tomorrow morning you could be speaking seven different languages.

Fluently.

Our family of true compilers will have your 680X0 Unix-based computer speaking COBOL, C, PASCAL, RPG II, PL/I, BASIC, and FORTRAN in the morning.

Not only can your customers take advantage of the power of your new systems, but they can do it without sacrificing their present software investment because we conform to ANSI and industry standards. And now we're 80386 compatible too.

Think of the markets this will open up for you. All for less than developing just one compiler yourself.

There's a lot more you should find out, so do it now.

We'll put you on speaking terms with the world.

Language Processors, Inc., 400-1 Totten Pond Road,
Waltham, MA 02154, (617) 890-1155.

LPI is a trademark of Language Processors, Inc.
Unix is a trademark of AT&T.Circle no. 266 on reader service card.

ever, needs a more realistic expression analyzer. One is presented here in Listing Six, page 64. If you don't know what a formal grammar is or how a recursive descent parser works, you should either go back and read the previous C Chest or look at my book *The C Companion*.⁶

The analyzer is called with:

```
VTYPE parse(expr_p)
char **expr_p;
```

VTYPE is currently defined to *double*, but you can change it to any type that's convenient and recompile if you wish. Changing to an integral type will make the program smaller. *Expr_p* is a pointer to a string pointer. The string itself holds the expression to analyze. The expression is parsed, **expr_p* is adjusted to point past the parsed string, and the result of the expression evaluation is returned. Evaluation terminates when the first character that can't be part of an expression is encountered.

Expressions are composed of:

spaces—All white space is ignored.

numbers—if *VTYPE* is a floating-point type (*float* or *double*), then numbers with decimal points are permitted; otherwise, a period is an illegal character and should be shunned.

operators—Several operators are supported, as shown in Table 3, right. Operators on higher lines have higher precedence. They all associate left to right and evaluate as in C. Unlike C, evaluation of *&&* and *//* does not terminate when the truth or falsity of the expression can be determined. The minus sign on the top line is unary minus; the one on line 3 is binary minus. The '*str1' str2'* operator works just like *strcmp(str1,str2)* does. Parentheses are for grouping rather than for subroutine calls.

The grammar used is shown on lines 27 to 56 of Listing Six. It is a classic LL(1) expression grammar. For the most part, the subroutines follow the productions quite closely. Note that I've merged several productions into a single subroutine when possible.

An example of this merging is shown in Code Example 3, below. A description of the merging process is in *The C Companion*, cited earlier.

Notes

1. A note on nomenclature. The two Unix text formatters, *nroff* and *troff*, are almost identical. The main differences have to do with commands that are unique in typesetting (point changes and so on) as compared to simple text formatting. *Nr* implements all of *nroff*, but the implementation of the *troff* features is spotty. For simplicity's sake, I'll say *nroff* throughout, even if I'm talking about what is really a feature of *troff*.
2. Brian Kernighan and P. J. Plauger, *Software Tools in Pascal* (Reading, Mass.: Addison-Wesley, 1981).
3. I'm using a version of the Unix *vi* editor called *PC/VI*. It's \$149 from Custom Software Systems, P.O. Box 678, Natick, MA 01760, and I recommend it highly if you're a *vi* addict like me. *PC/VI* is a solid and very complete implementation of *vi* for MS-DOS.
4. Hash algorithms are discussed in greater depth by Robert Kruse, *Data Structures and Program Design* (Englewood Cliffs, N.J.: Prentice-Hall, 1984), pages 112–128, and also by Aaron Tenenbaum and Moshe Augenstein, *Data Structures Using Pascal*, 2d ed. (Englewood Cliffs, N.J.: Prentice-Hall, 1986), pages 521–573. Both these books are very good. The examples, all in nicely done Pascal, are informative and useful.
5. Aho, Sethi, and Ullman, *Compilers: Principles, Techniques, and Tools* (Reading, Mass.: Addison-Wesley, 1986), 436.
6. Allen I. Holub, *The C Companion* (Englewood Cliffs, N.J.: Prentice-Hall, 1987), 189–211. The descriptions in the *Companion* are a bit better than those in the original C Chest articles.

DDJ

(Listings begin on page 52.)

Vote for your favorite feature/article.
Circle Reader Service No. 5.

```
() - ! 'str1' str2'
* / %
+ -
< < = > > = == != =
&&
```

Table 3: Operators supported by parse()

The productions:

```
<fact> ::= <part> <fact1>
<fact1> ::= + <part> <fact1>
           ::= - <part> <fact1>
           ::= epsilon
```

can be implemented as:

```
static VTYPE fact()
{
    VTYPE left;
    left = part();
    for(;;)
    {
        if (match("+")) { advance(1); left += part(); }
        else if (match("-")) { advance(1); left -= part(); }
        else break;
    }
    return left;
}
```

Code Example 3: Merging two productions into a single subroutine

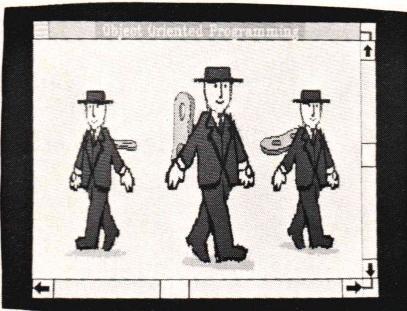
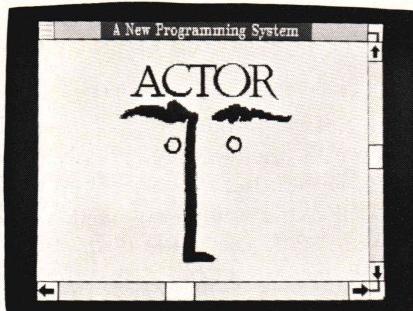
ACTOR WILL DO WINDOWS.

If you've been wondering how you're going to explore Microsoft® Windows and then work it into your programs, what you need has just arrived.

ACTOR™

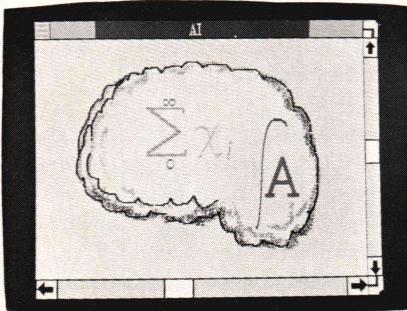
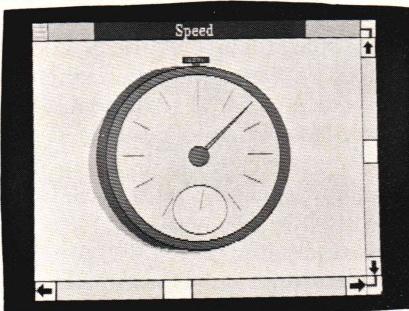
ACTOR is a new programming system, and there are two reasons why it's so good with Windows.

First, ACTOR is an ideal programming environment. It shows you all your work through Windows. Any part of the program you're writing, its output as it runs, error diagnostics and tracing, catalogs of routines, and, in fact, just about anything you like—they can all be visible on the screen and accessible at the same time, in their own windows.



ACTOR™ is a new, interactive programming system, the first with Microsoft® Windows. You have Windows when you write a program, and users have Windows when they run it.

Moreover, ACTOR is a new, object-oriented programming language. Wind-ing objects up and turning them loose is a lot easier, more productive, and more fun than old fashioned programming.



Thanks to a new method of "incremental garbage collection," ACTOR never has to slow down to clean up memory like other advanced languages. You can even use it for real-time control. It's that fast.

ACTOR offers all the features of an ideal artificial intelligence language, but in a familiar, Pascal-style syntax. Which makes artificial intelligence programming easier. And what could be smarter than that?

that windows are objects like everything else in the language. You make as many as you need, and they do what they're supposed to, mostly on their own.

So it's natural and easy to put windows in your application programs. And users will benefit from Windows when your programs run, as much as you do when you write them.

To order, just send in the coupon. Or call **312-491-2370** for more information. You'll see that ACTOR will do Windows, and more.

Please send me more information about ACTOR.

Name _____

Address _____

City _____ State _____ Zip _____

Daytime phone _____

Check or money order enclosed. Bill my VISA. Bill my MasterCard.

Card number _____ Exp. date _____

I'd like to place an order while I'm at it. (ACTOR is not copy protected and has a 60-day money back guarantee. Educational pricing is available. ACTOR requires an IBM PC/XT or AT or equivalent. 640K and hard disk is recommended.)

copies of ACTOR with Windows runtime support at \$495 each. \$_____

copies of ACTOR, including Windows complete with MS-DOS Executive, at \$550 each. \$_____

Please add \$5 for shipping within the U.S., \$30 overseas. \$_____

Total. \$_____



The Whitewater Group®

Technology Innovation Center
906 University Place, Evanston, IL 60201

16-BIT SOFTWARE TOOLBOX

Resources for MS-DOS Programmers

Jourdain, Robert. *Programmer's Problem Solver for the IBM PC, XT, and AT*. New York: Brady Communications, 1986. 473 pages with index. \$22.95. ISBN 0-89303-787-7

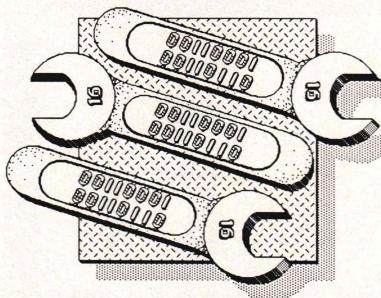
Topics covered include determining the system configuration; managing interrupts; memory allocation; reading and setting timers; controlling video adapters; creating tones; controlling the keyboard interface, printer, and serial port; reading and writing disks; and device drivers. For the most part, each topic is accompanied by three example program listings: a high-level routine in BASIC, an intermediate-level routine in assembly language that calls the operating system or ROM BIOS, and a low-level routine in assembly language that accesses the hardware directly. Almost anything you can imagine wanting to do to, with, or on an IBM PC can be found in this book, including reading and writing files to a cassette recorder! But the book is not a tutorial and will be most useful in combination with a book such as *The Peter Norton Programmer's Guide to the IBM PC*, Angermeyer and Jaeger's *MS-DOS Developer's Guide*, or my own *Advanced MS-DOS* (yes, that was a plug).

Rollins, Dan. *IBM-PC 8088 Macro As-*

by Ray Duncan

sembler Programming. New York: Macmillan, 1985. 435 pages with index. \$25.50. ISBN 0-02-403210-7

A very nice primer on 8086/88 assembly language, starting at the most elementary level but progressing to advanced topics such as structures, macros, and conditional assembly. It has a brief section on file and record



operations under MS-DOS, but this only covers the now-obsolete file control block functions. The book ends with a detailed discussion of programming for the IBM PC video interface, including graphics modes, that is helpful and practical.

Sargent, Murray, III, and Shoemaker, Richard L. *The IBM PC from the Inside Out*. rev. ed. Reading, Mass.: Addison-Wesley, 1986. 483 pages including index. \$19.95. ISBN 0-201-06918-0

This book covers a wide range of topics and is the only MS-DOS book I know of that is directed at the hardware hacker. The PC system bus, peripheral chips, and controllers are discussed in great detail with many programming examples, and the book winds up with a chapter on how to breadboard your own interfaces. It's a real pity that Addison-Wesley didn't see fit to invest in decent production for this book, especially as it was popular enough to warrant a second edition; instead, the book was offset from camera-ready copy prepared on a daisy-wheel printer with a particularly crowded and tiring sans serif font.

King, Richard Allen. *The MS-DOS Handbook*. 2d ed. Berkeley, Calif.: Sybex Inc., 1986. 338 pages including index. \$19.95. ISBN 0-89588-352-X

I discussed the first edition of this book in my May 1986 column. The second edition covers essentially the same ground but has some additional

material on networking and MS-DOS, Version 3. Incidentally, previously I commented that this book "jumbled together" material about MS-DOS and PC-DOS "with very little distinction." In a letter that accompanied the second edition, Mr. King said, "You are right, . . . [but] my claim is that it does not matter, and few people make the distinction anyway." The year that has passed since I originally wrote those carping words has vindicated Mr. King. The incredible domination of the marketplace by the IBM PC architecture has made any distinction between generic MS-DOS and IBM versions of MS-DOS a moot point.

DeVoney, Chris. *Using PC-DOS*. Indianapolis, Ind.: Que Corp., 1986. 519 pages including index. \$21.95. ISBN 0-88022-170-4

This is not a programming book but is a superuser manual to PC-DOS that eclipses all other such books, including all the Microsoft and IBM manuals. The last section of the book is a thorough, alphabetical reference to PC-DOS commands, including version dependency information and a detailed list of error messages.

The MS-DOS STACK Command

People who have switched from earlier versions of MS-DOS to Version 3.2 have sometimes been surprised to find that previously healthy software caused the system to halt with the mysterious message "Internal Stack Error."

It turns out that the boys from Boca got concerned because when network cards were active a great many interrupts could occur in rapid succession, causing a program's stack to overflow with consequent unpredictable damage to the system. They

therefore instigated a new scheme in MS-DOS 3.2 such that when an interrupt occurs, the system automatically switches to a stack allocated from an internal pool before passing control to the handler. When the interrupt service is complete, the stack is released back to the pool. When sufficient interrupts occur in a brief period and are all simultaneously in various stages of processing, the stack pool can be exhausted and the system halts with the previously mentioned error message.

The number and size of stacks in the internal pool can be controlled at system initialization time by adding the new MS-DOS 3.2 *STACKS=n,s* command to the CONFIG.SYS file, where *n* is the number of stack frames (8–64, with a default of 9) and *s* is the size (in bytes) of each stack frame (32–512, with a default of 128).

Remember that this "feature" was specifically added to IBM's version of MS-DOS and may or not be present (or documented) in other OEM versions of MS-DOS.

Boyer-Moore Algorithm

Heartfelt thanks to all the readers who wrote in with comments, explanations, and improvements on the Boyer-Moore routine published in the October 1986 16-Bit Software Toolbox column. By an interesting coincidence, the November 1986 issue of *Computer Language* carried a lengthy article on the same subject. Frankly, I am still digesting that opus and all the feedback I got on the subject from *DDJ* readers, so I will defer a resumption of this discussion until the future.

TSRs and File I/O

Terry Flanagan, of GSD Development Corp., Chicago, Illinois, responded with the following letter to Gary Cramblitt's appeal, printed in the September 1986 16-Bit Toolbox column, for help with Terminate and Stay Resident utilities: "The problem is, as Gary mentioned, that MS-DOS is not reentrant. It is a single-user operating system, and as such, reentrancy is not a requirement. This situation severely limits the processing that can be performed by a memory-resident program, however."

"The problem has been resolved to some extent by various software com-

panies that have developed memory-resident utilities. There are still conflicts between these various utilities, and there is still no officially sanctioned method for providing compatibility between memory-resident programs. A multitasking version of MS-DOS will probably appear before a TSR standard is adopted. During the interim, developers of memory-resident programs are on their own.

"There are several methods I've read about and/or experimented with to determine when MS-DOS is 'safe.' One method, which Gary mentioned in his letter, is the DOS Critical Flag. This is a byte within the DOS code segment that is incremented upon entry to the DOS function dispatcher (*int 21h*) for any potentially 'unsafe' function calls and decremented upon exit. The function that returns the address of that flag is *int 21h*, function *34h*. The DOS Critical Flag, therefore, is a counter that indicates the number of DOS calls in progress. Normally, this counter never exceeds 1. So it is theoretically safe to call DOS whenever this counter is 0. Unfortunately this is not usually the case.

"Certain DOS calls are relatively safe to interrupt. These are the character I/O calls (*int 21h*, functions 1 through *0ch*). DOS normally resets its own internal stack for each function call. If a character I/O call has been interrupted, however, as indicated by an internal flag, then DOS will adjust the stack pointer upward so as not to collide with the stack of the character I/O call in progress. The problem is how to determine when character I/O is in progress.

"Borland's SideKick is a pioneer in this area. Borland decided to play it safe by letting SideKick run off with as many interrupt vectors as it could carry. This is typical of many memory-resident utilities and results in the utilities stealing vectors from one another and causing further confusion. SideKick even uses the timer tick to steal back the interrupt vectors. (Sooner or later someone will just reprogram the 8259 to hide the real interrupt vectors from everyone else and really mess things up.)

"Anyway, one of the vectors that SideKick 'borrows' is the DOS function dispatcher (*int 21h*). By intercepting the DOS calls the program can determine when it is safe to call DOS

from the memory-resident code. The program can also block access to DOS or reschedule DOS calls if desired.

"Another method for determining when DOS calls are safe involves *int 28h*. I've read that this interrupt occurs while COMMAND.COM is waiting for keyboard input and DOS is basically in an idle state. The DOS background print processor supposedly uses this interrupt to perform print operations.

"From what I've seen, this interrupt is called from DOS prior to nondestructive reads to the console to check for things such as Control-C. This would normally occur during character I/O calls. Prior to calling *int 28h*, DOS pushes a byte flag on the stack (really a word, but only the low-order byte matters) that indicates whether a 'safe' character I/O call is in progress or not. This byte contains a 1 if the call is safe; otherwise it is 0. So it would seem that if the DOS Critical Flag is 0 or the Critical Flag is 1 and the byte pushed before the *int 28h* call is 1, then it would be safe to call DOS. I'm not sure I would even want to think about what happens when a 'critical error' occurs on a DOS call that interrupts a character I/O call, though.

"In addition to determining whether DOS is safe or not, a memory-resident program must be reactivated somehow. Most of the currently available utilities use the concept of a 'hot key' to trigger program processing. The hot key can be detected by monitoring keyboard input through the hardware interrupt (*int 9*) and/or through the keyboard BIOS driver interrupt vector (*int 16h*). Some programs use a combination of these methods. Some utilities also use the timer interrupt (*int 8*) to monitor the state of the keyboard shift flags if the hot key involves a shift-key combination.

"Once the hot key is detected, the program can wait for DOS to become safe. This usually involves setting flags and then using the timer interrupt to trigger hot-key processing, although there are numerous possibilities here. For Gary's screen dump it would probably be advisable to save the screen in memory when the hot key is detected, then dump it as soon as DOS is safe.

"One other point that may be worth mentioning deals with the PSP

(program segment prefix). The PSP is used to store things such as the environment pointer, command tail, file handles, caller's stack pointer during DOS calls, and so on. It is also the DOS equivalent of a process ID (PID). When a memory-resident program is reactivated, DOS is still assuming the PSP of the currently executing program. Therefore it is advisable to switch to the PSP of the memory-resident program [to ensure that MS-DOS is looking at the correct table of file handles if the memory-resident program needs to perform file I/O—Ray]. At installation time, the TSR should use the DOS function call *62h* (MS-DOS 3.0 and later) to obtain its PSP address and store it for future reference (versions of DOS prior to 3.0 should use the undocumented function *51h* to do the same). When the memory-resident program is reactivated, it should use another undocumented function (*50h*) to change the PSP and restore it before exiting. These are safe, if undocumented, DOS calls.

"There are no guarantees for any of these methods, especially those involving undocumented DOS calls (why does Microsoft keep these a secret anyhow?). I hope that some sort of standard for TSR programs will be developed and that these mysteries will be solved in the near future."

It is true, as Terry notes, that the MS-DOS calls he mentions are undocumented and therefore not officially supported. It is my feeling, however, that it is quite safe to use these calls in

MS-DOS, Versions 2 and 3, because these versions of the operating system seem to be quite stable and it does not appear that DOS 4 will ever be released for 8086/88 machines in the United States. The calling sequences for these MS-DOS functions are in Table 1, below.

On the subject of conflicting Terminate and Stay Resident utilities, there was a great flurry of press releases about a year ago to the effect that Microsoft, Borland, and other worthies

| MS-DOS Call | Call With | Returns |
|--|--|--|
| <i>int 21h, function 34h*</i> Get DOS Critical Flag address | <i>ah = 34h</i> | <i>es:bx =</i> address of DOS critical flag |
| <i>int 21h, function 50h</i> Set current PSP | <i>ah = 50h</i> <i>bx = PSP (segment) address</i> | nothing |
| <i>int 21h, function 51h**</i> Get current PSP | <i>ah = 51h</i> | <i>bx = PSP (segment) address</i> |

* Interestingly, function *34h* is documented in the *Heath/Zenith Version 2 Programmer's Utility Pack* manual but is absent from all other OEM editions of the *MS-DOS Programmer's Reference*.

** Apparently identical to *int 21h, function 62h* in MS-DOS 3.0 and later (so why give it another function number, rather than just document function *51h*? Another MS-DOS mystery . . .)

Table 1: Undocumented MS-DOS calls 34h, 50h, and 51h

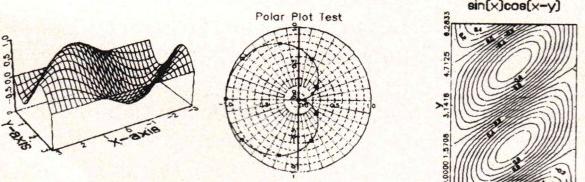
Publication Quality Scientific Graphics

Over 100 C routines make scientific plotting easy

- linear, log, & polar plots
- bar charts & Smith charts
- contour plots with labels
- 3-D curves, 3-D surfaces
- 4 curve types, 8 markers, errorbars
- 14 fonts, font editor
- unlimited levels of ^{sub}scripts
- 4096 × 3120 resolution in 16 colors on EGA, Tecmar, Sigma boards
- zoom, pan, window and merge plots
- high resolution printer dumps

SOURCE INCLUDED for personal use only
\$350. Demo \$8

256k, IBM, AT&T, Corona PCs, DOS 2.xx, 3.xx
Most boards, printers, and plotters supported
Microsoft, Lattice, DeSmet, Aztec, C86 compilers



Scientific Endeavors Corporation
Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

Circle no. 210 on reader service card.

9-Track Tape Subsystem for the IBM PC/XT/AT

XENIX or MS-DOS.

The solution to your micro/mainframe communications problem is available today!



Qualstar's new $\frac{1}{2}$ inch 9-track MINISTREAMER™ brings full ANSI data interchange capability to the PC. Now you can exchange data files with virtually any other computer using 9-track tape.

Available in both 7" and 10 $\frac{1}{2}$ " versions, the MINISTREAMER uses less desk space than an ordinary sheet of paper, yet provides full 1600/3200 BPI capability at an affordable price. Up to 134 megabytes of data (depending on format) can be stored on a standard 10 $\frac{1}{2}$ " reel of tape, thus making the MINISTREAMER a highly-reliable answer to your backup requirements as well.

Tape subsystem includes tape drive, full-slot coupler card, cables, dust-cover and MS-DOS or XENIX compatible software.



Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!

QUALSTAR®

9015 Eton Avenue,
Canoga Park, CA 91304
Telephone: (818) 882-5822

Circle no. 356 on reader service card.

were going to work out and cosponsor a TSR specification that would solve these problems forever. No word of progress on this document has been forthcoming for the last six months, however, and it appears to have died the lingering and painful death of neglect. The torch has been taken up by a small committee that includes Chip Rabinowitz, Chris Dunford, Jim Kyle, Neil Rubenking, and Lane Ferris, who have been developing

a public-domain protocol for TSRs called Ringmaster. The Ringmaster effort has been quietly gathering support and seems likely to have become the de facto standard by the time you read this.

Removing Extra EOF Characters

Robert Seabock, of Tamal, California, wrote: "Concerning your report [September 1986] of that bizarre bug in

Version 4 of the Microsoft Macro Assembler [where extra EOF characters cause problems in include files] and Microsoft's suggested 'fix': For those people (myself included) who would rather pick up a skin rash than a copy of EDLIN, the easiest way to remove EOF characters from the end of a text file is to use the MS-DOS TYPE command, redirecting output to a file. For example:

```
TYPE myfile > temp
DEL myfile
REN temp myfile
```

would do the job quite nicely."

Thanks Robert, you are right. But to my chagrin, Robert Thrun of Adelphi, Maryland, pointed out that there is an even easier way to strip off those nasty extra EOF characters, to wit:

```
COPY myfile /A newfile
```

The /A (for ASCII) switch is present in the COPY command just for this type of problem! It copies a file only up until the first EOF mark, then stops.

Code Example 1: The difference between MASM's equ and = operators

FTL MODULA-2 \$49.95

*The most programming power
for your money*

- FTL MODULA-2 gives you:
- Full implementation of MODULA-2.
- Split screen, multi-file editor.
- Sources to the run-time modules.
- Complete support for the product.
- 8087 reals available for MSDOS.*

FTL MODULA-2 meets the standards in Niklaus Wirth's book, "Programming in MODULA-2", third edition. No other language gives you better flexibility, code design, or ease of code maintenance.

FTL MODULA-2 gives you compact, rommable code quickly. This is the language of the future. If you program in Pascal or C, FTL MODULA-2 will increase your output per programming hour. If you want to learn high level structured language programming, this is the best starting point.

FTL MODULA-2 was named "product of the year" by Jerry Pournelle in his column, "Computing at Chaos Manor", BYTE Magazine, April, 1986.

* 8087 support is an addition \$39.95

FTL Editor toolkit \$39.95

The source code to the editor is offered as a toolkit. The editor was written in FTL MODULA-2 and this set of programs provides the novice with working sources to speed up learning. The experienced programmer will find the wealth of pre-written modules a time-saving bonanza.

SAVE \$10.00

Buy both the compiler and the editor toolkit for only \$79.95. That's a \$10.00 savings off the regular price. FTL MODULA-2 is available for both MSDOS and CP/M-80 systems.



SEND
CHECK OR
MONEY ORDER
TO:
**Workman
& Associates**

1925 E. Mountain Street
Pasadena, CA 91104
(818) 791-7979 voice
(818) 791-1013 data M-F 8pm-8am
Or join us on BIX (Byte Information eXchange
W.A.N.D.A. Conference)

MasterCard and VISA welcome.
Please add \$3.00 Shipping and Handling.
Calif. residents please add 6% sales tax.

Circle no. 244 on reader service card.

Clarify your Source Code

C, BASIC, Pascal, dBase, Modula-2 programmers: Be more productive with two new utilities from Aldebaran Laboratories!

Source Print™ organizes your source code, simplifies debugging, makes documentation a snap

- Index (cross reference) for variables, functions, procedures, fields
- Structure outlining to draw lines around nested structures
- Automatic indentation
- Table of contents listing functions, procedures, subroutines
- Boldface printing of key words
- Split multistatement BASIC lines for readability
- Extract routines by name



\$75.

Tree Diagrammer™ identifies the hierarchical structure of your program

- Prints organization chart of program automatically
- Illustrates hierarchy of calls to functions, procedures, subroutines
- Indicates recursive calls

\$55.

Both utilities have an easy-to-use menu with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Get your programs organized now. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee.

800-257-5773, 800-257-5774 (CA)

MC, VISA, AX, COD. Add \$5 for shipping.

Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBase is a trademark of Ashton Tate.
Handles up to 50 files, 60,000 lines.

Aldebaran Laboratories, 3339 Vincent Rd.
Pleasant Hill, CA 94523 415-930-8966

YES! Rush me

Source Print @ \$75. _____
 Tree Diagrammer @ \$55. _____
Ship/Handling \$5. For CA add 6% tax _____
TOTAL _____
Name _____
Company _____
Address _____
City _____ State _____ Zip _____
 Check enclosed VISA Mastercard American Express
Card # _____ Exp. Date _____
Signature _____ Phone # _____

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

—PC Magazine
Sept. 16, '86

16-BIT

(continued from page 105)

Macro Assembler Equates

Microsoft MASM's *equ* and = operators are used to associate a value or address with a name and facilitate the creation of readable, well-documented, maintainable assembler code. There is a subtle distinction between these two operators that is not obvious from the manual but that can nail you at unexpected times! The difference is that *equ* basically creates a text macro, whereas the = operator is evaluated when it is encountered and assigns an absolute value to a symbol. This distinction is perhaps best illustrated by a simple example. (See Code Example 1, page 105.)

In this example, I've assigned the expression *offset (\$)* - *offset test* to two different labels created with *equ* and =. As you can see, the symbol *test_equ* created with *equ* is evaluated when it is invoked, in this case resulting in a different value each time it is used. The symbol *test_equals* created with = is evidently evaluated at the time it is declared, so it has a constant value.

Lay Down Your Pencils

Hans Pufal's pop quiz in the September 1986 16-Bit Software Toolbox unfortunately went somewhat awry because of my own bad eyes and errant typing. Hans' original code was correct, but a typo in the printed listing led a number of readers to send comments along the lines of "the code doesn't do what its author thinks it does."

The real answer is that the little routine converts a hexadecimal nibble in register *al* to its ASCII character equivalent, leaving the result in *al*. The corrected code appears in Code Example 2, below.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 6.

```
and al,0fh
add al,90h
daa
adc al,40h
daa
```

Code Example 2: Corrected code for Hans Pufal's pop quiz

Circle no. 350 on reader service card.

C spoken here...

High C™

Do you want to use a C compiler that

- was chosen by Ashton-Tate for dBASE III® Plus, and CAD Leader AutoDesk for AUTOCAD and AUTOSKETCH "with a twenty percent code savings over Lattice C."
- was well rated in **Computer Language, Feb. 86:** "Then there is High C, the most powerful compiler of all..." and **Dr. Dobbs Journal, August 86**
- "would have saved me three weeks of porting time had I had High C instead of Microsoft's new C"
- Mike LeBlanc, compiler developer, Sky Computers*
- "is the only C compiler for the IBM PC capable of compiling NYU's Ada/Ed compiler"
- Dave Shields, research scientist, New York Univ.*
- has a complete run-time library
- has structure assignment, **enum, void...**
- supports nested functions as in Pascal
- supports pcc and full K&R C plus some latest, nifty extensions from the new ANSI-proposed C standard
- "is the highest quality C compiler for large-scale software development." *Randy Nielsen, Ansa (Paradox)*
- "saved 15% of code over five large modules of MultiMate relative to Lattice C"

David Beauchesne, Multimate International

Power Tools

Each compiler • generates **superb code**, with optimizations such as common-subexpression elimination and cross-jumping • sports no less than **five memory models** for the 8086 (Small, Compact, Medium, Big, and Large) • supports a unique implementation of register variables • supports the **8087/80287** in native mode, or **emulates** • supplies **three floating-point formats** • generates special instructions for the 80186/286 • generates code that runs in **80286 protected mode** • gives you **hundreds of error and warning messages**, helping you find those subtle bugs *before* you need a debugger's help • lets you **overlay data** as well as code (when used with PLINK86), for substantial space savings • lets you write **interrupt routines** directly in high-level language • lets you get "close to the machine" with built-in move/scan/compare operations • is supported by equivalent **resident and cross compilers** for the 80286 (UNIX V.2, Xenix, Concurrent DOS 286), 80386 (DOS, UNIX V.3), 68010/20/68881 (UNIX V & 4.2, GEM-DOS™), 32032 (UNIX 4.2), VAX (UNIX 4.2, VMS), RT PC, IBM 370,... • contains a **multi-modular cross-referencer** • produces ROM-able code for embedded applications • can talk to those **other languages** by those other vendors • gives you **64K run-time stack** space that can be shared with the heap • is endowed with an **amazing number of pragmas** (compiler controls) for customization to your application • has a compiler start-up **profile** • supports direct access to MS-DOS; library supports DOS 3.X file-sharing • generates symbolic debugger information for use with all known MS-DOS debuggers • allows **exec-ing** subprocesses • was designed for professional software developers, not hobbyists • comes with great technical support by a company that specializes in compilers • comes with extensive **typeset documentation** • and more... call or write for your information packet today...

Not recommended for casual use, but for applications needing **industrial-strength tools**, contact



Meta **Ware**™
INCORPORATED

903 Pacific Avenue, Suite 201, Santa Cruz, CA 95060-4429
(408) 429-6382 (429-META), TELEX: 493-0879

High C V1.3: \$495

OEMs: Contact us about porting our professional compilers to your systems.

TWS: Professional Compiler Developers and competitors, ask about our Translator Writing System compiler toolbox; see the review in **Computer Language, December, 1985**.

DOS Helper™: Powerful UNIX-like utilities to enhance MS-DOS, for \$49.95—included free with MetaWare compilers: FIND, TAIL, MV, LS, CAT, UNIQ, FGREP, and WC.

C Validation Suite: Used by some of our competitors and many others.

Professional Pascal V2.6: \$595

\$2,000/Plant Site

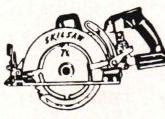
• MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated • Other trademarks and their owners are: UNIX—AT&T, dBASE III—Ashton-Tate, Volkswriter Deluxe—Lifetree Software, GEM-DOS—Digital Research, Ada-DoD. © 1986 MetaWare Incorporated.

Pascal spoken here...

Professional Pascal™

Do you want to use a Pascal compiler that

- was chosen by Lifetree Software, Inc., for implementing Volkswriter Deluxe™
- serves as a systems and applications language at CAD/CAM giant Daisy Systems Corporation
- was well rated in **Computer Language, May 86:** "The clear choice for large-scale programming projects..." and **PC Tech Journal, July 86:** "for team programming or for very large projects...stands absolutely alone." pg. 126 "documentation is certainly the best...a model of technical clarity." pg. 112.
- is the "howitzer" of Pascals and "could well be the most powerful Pascal compiler ever implemented on a microcomputer" **PC Magazine, Oct. 29, 1985, p. 144**
- has 8-, 16-, and 32-bit integers; sets up to 64K bits
- has varying-strings of up to 64K characters
- has a full-fledged C macro preprocessor
- has many run-time library additions: UNIX™-like I/O, multiple heaps, interrupts, . . .
- has all the bit-pushing operators of C
- has many more extensions, getting you half way to Ada® for a non-Ada price



for

Power Users

Abroad:

ABC Software, The Netherlands
Microsoft, Tokyo
Grey Matter, United Kingdom
Buchdata, Frankfurt

Professional Tools Since 1979

What Progress Is Being Made in AI?

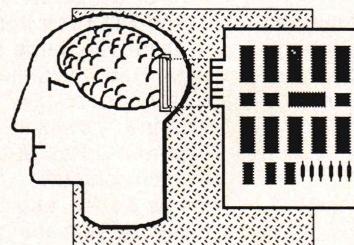
As this is the first of the monthly columns I will be writing on artificial intelligence, it seemed that a good way to begin would be to give a progress report on the state of the art in AI. Much of what I will report of a factual nature is based on a close monitoring of the newest developments in the field, especially as demonstrated at the main AI conferences of the last few years—the International Joint Conference on Artificial Intelligence (IJCAI) in Los Angeles in 1985 and the AAAI conference in Philadelphia in August 1986. Unless otherwise stated, of course, the opinions and extrapolations are solely my own.

It may well be true that AI still has a long way to go to overtake all the hype, but if you have been watching closely over the last two years or so, you may have noticed that the field has been making some interesting progress each year. At the 1986 AAAI conference, for example, there was a noticeable presence of some serious AI applications, not just more and better tools. The main mood of the field at this time seems to be that of proving its usefulness in as many legitimate areas as it can. Some powerful development tools have been available commercially for a few years, and an impressive and diverse array of applications has begun to appear.

Many programmers recognize instinctively that they can learn something from AI that will make them

by Ernest R. Tello

better programmers, even if they are not planning to do work specifically in the AI field. One common misconception, though, is that the most they can learn is some new techniques to add to their arsenal of programming devices. AI has much more to offer than this. You will be hearing a lot



about programming paradigms both in my monthly column here and elsewhere, so this is a good time to explain the difference between programming techniques and programming paradigms.

One thing all people in AI agree about, regardless of how vehemently they may disagree on other matters, is that we still do not know how to program computers very well—there are many things still to be learned that can greatly improve our programming. Another thing that is generally agreed upon among people in the AI field is that nearly all conventional programming comes under one major paradigm, namely that of procedural programming. Once this is pointed out, it is almost unnecessary to dwell on the fact that for most programmers designing a program simply involves planning procedures for the computer to execute. As things stand now, though, procedural programming is just one of many paradigms at the disposal of AI programmers.

The object-oriented paradigm represents another approach entirely. Arising originally through AI research, and currently one of the most popular of all the paradigms in AI, it is just now penetrating the commercial programming market. The rule-based programming paradigm has also entered the commercial mainstream, mainly through its successful use in the hot field of expert system technology. The logic programming paradigm as well as the declarative and rule-based programming paradigms were all combined to create

the powerful new programming language PROLOG, another recent newcomer. Other less-known programming paradigms that have also originated in AI research include constraint propagation, access-oriented programming, the actor paradigm, the neural network paradigm, and various parallel and distributed programming paradigms such as the connectionist and data-flow approaches. As you can see, the field of AI today is large and complex, but some clear trends show a definite unity amid the diversity.

Another good thing for programmers to know is that there are several clear indications that, for better or worse, the goals of AI are on the verge of undergoing a paradigm shift. According to Mark Stefik at Xerox PARC, AI is now in the process of becoming the latest vehicle for the development and proliferation of a new form of "knowledge media." Knowledge media, according to Stefik, embrace not only the representation and storage of knowledge but also its transmission throughout society. At this point, at least, AI systems are not a major medium for the transmission of knowledge. As AI technology matures, though, Stefik argues that it will increasingly take on this role as a carrier and transmitter.

It is obvious that AI systems already differ from most other knowledge media such as books, diagrams, and films in that they are not entirely frozen and inert. The knowledge in an AI system is dynamic and can be executed for solving problems in a more flexible way than is possible with any other knowledge medium. In this respect, it already resembles humans. But, so far, the ways in which AI systems can use knowledge to solve problems is still extremely limited as compared with humans.

Has AI given up on the quest to create machine intelligence as an arti-

NEW

COMMON LISP Development System for Your PC or AT

Introducing TransLISP PLUS™, The Consultant's LISP

Over 400 Primitives, a C Interface, and Optional Runtime System

People call you because you're an expert. Your customers want to keep up with what's going on. They want software that is more intelligent and responsive, or software that does something that just wasn't possible before. So they call you.

Now you can write and deliver a whole new category of software with TransLISP PLUS, a practical, efficient LISP system. You can also add Artificial Intelligence to your software. We include several features, like a C Interface and Optional Runtime System, so you can control the performance and security of your programs. And your users only need to have a PC (can even be non-compatible) with 320K and 1 floppy drive. Now, what could you write for a \$700 PC? . . . or a \$7000 PC? . . .

Add AI Technologies to Your Software

Most of the programs you write are accessed by a user who doesn't have your expertise. Use intelligent interfaces to make your programs more responsive to the end user.

You can even use the C Interface included with TransLISP PLUS to customize LISP, or combine C functions with LISP programs.

Take advantage of AI technologies to make your programs smarter and more flexible.

Extensive Development Environment Over 400 Primitives

TransLISP PLUS provides you with over 400 primitives for development, including extras for hardware support and operating system access. Their spectrum ranges from control constructs, macros, and special forms, to multi-dimensional arrays, reader support for binary, octal, and hex constants, improved list processing, and system interrupts.

DOS commands and applications can be invoked from within TransLISP PLUS, as can the fast editor. Of course, you can use your own editor if you like.

A variety of debugging tools are provided. The trace facility tracks the evaluation of any built-in or user-defined function or macro.

Traceback, Break, Cross Reference, and Pretty Printer are also provided to help you spot problems.

- Over 400 COMMON LISP Primitives
- Optional Runtime (No Royalties)
- Interpreter
- Program Editor
- Many Debugging Utilities
- Microsoft Mouse Support
- Supports IBM PC color graphics
- Supports 8087 math coprocessor
- Over 30 Demo Programs with Source

The ONLY Full Featured Common Lisp with a C Language Interface

The best of both worlds. The interface to Microsoft C gives you a powerful extension to TransLISP PLUS — now you can write code in LISP and C. And you don't need an AT, it will run on your PC!

The C Interface makes it practical for you to write a C program and add it as a new function to TransLISP PLUS. Your function can:

- extend and/or change the LISP syntax
- be an entire system of programs

Create your own BUILT-IN primitives which are directly tied to the system and called at full speed by the interpreter. Extend the functionality of your program by including features of your own like macros, functions, and special forms.

Code from C libraries produced by other vendors can be integrated into your program to perform tasks not normally part of LISP.

Use PLUS for Your Applications. No Royalties.

Once you own TransLISP PLUS, you may want to use it to distribute applications. No problem.

The Optional TransLISP PLUS Runtime supplies you with a special interpreter. You can distribute an executable version of your program without distributing source code.

The Runtime is available for \$150 and TransLISP PLUS is required.

- C Language Interface
- Complete Manual with Tutorial, Indexed Reference Manual, and Quick Reference Card
- Online Help
- Lexically scoped
- Use your C Libraries
- System Interrupts
- NOT COPY PROTECTED

Use TransLISP PLUS to program with and deliver to lots of machines . . . Use your existing Libraries . . . Distribute your applications

MONEY BACK GUARANTEE

Try TransLISP PLUS (\$195) for 30 days — if not satisfied get a full refund.

Circle no. 148 on reader service card.

Don't Know LISP?

Get a solid understanding of LISP with the comprehensive, easy-to-understand tutorial. Each section walks you through a new concept and reinforces it with examples — both text and online.

Over 30 demo programs supplement the tutorial. Use them for an in-depth introduction to LISP programming techniques. Commented source is included so you can see how and why the program operates.

The demos cover a wide variety of applications including: Select a word processor, Read dBASE SDF files, Job Counselor, and many, many more.

The Fundamentals

If you are interested in learning LISP but don't need all the extras in TransLISP PLUS, order TransLISP for \$95. It's a full, easy-to-use introduction to LISP that includes the tutorial and demo programs described above, and over 300 primitives.

It is a solid subset of COMMON LISP; and you can write programs of up to 12000 lines.

COMMON LISP Standard

Programs written carefully with TransLISP PLUS will be completely "portable" to any other COMMON LISP system on a micro, mini, or mainframe computer. This allows you, for example, to write a program with TransLISP PLUS on your PC at home, and compile and run it on the VAX at work.

System Requirements

TransLISP PLUS requires at least 320K RAM and a 360K disk drive.

TransLISP requires 256K RAM and a 360K disk drive.

| | |
|----------------------------------|-------|
| TransLISP PLUS | \$195 |
| TransLISP | \$ 95 |
| Upgrade TransLISP to PLUS . . . | \$158 |
| TransLISP PLUS Runtime | \$150 |

TO ORDER OR FOR DETAILS CALL

 **800-821-2492** 

**Solution
Systems™**

335-D Washington Street, Norwell, MA 02061, (617) 659-1571

TransLISP and TransLISP PLUS are trademarks of Solution Systems. Solution Systems is a trademark of Solution Systems.

fact, then, and settled upon the far easier goal of machine knowledge? This is both a crucial and yet extremely difficult question. Part of the problem is that, for the short term at least, knowledge appears to be far more useful than is intelligence. This may sound surprising, but it is important to remember that human intelligence must undergo a long training period before it can be trusted with any serious responsibilities. Just imagine buying a program that you had to teach for 16 years before it could do anything useful! But a knowledge-based system, even though it does not have much true intelligence, can do some useful things, usually without any training at all. All the same, I think that, in planning long-range projects and even shorter-range ones, to lean on knowledge too heavily and ignore the issue of how to make the processing of knowledge more intelligent is tantamount to sidestepping one of the most challenging issues AI faces.

The field of AI today is like a cell in which mitosis has occurred so that there are now really two independent bodies, both of whom seem to be thriving. I am referring to the two related fields of commercial applications of AI and basic AI research. I intend to give both some representation here because I think each of them are important and interesting.

Although some noteworthy progress is being made in AI today, I think it is fair to say that the field suffers from extreme fragmentation. A few major figures see the "big picture" and have a vision of sorts, but there still seem to be far too many researchers who see only the trees (sometimes just the leaves on the trees) and never the forests. So far, it has been primarily the Defense Advanced Research Projects Agency (DARPA) that has tried to impose some type of an overall agenda on AI research, but for several reasons, this has not and cannot succeed. The changes have to come from within AI itself. A new vision based on realistic long-range objectives and a scenario for its realization will have to emerge.

In spite of the lip service that is frequently paid to emulating the way

the human brain works, many AI researchers, even those who express great sympathy for this approach, do not, in my opinion, make a serious attempt to design systems that show an appreciation of the high-level organization of the brains of even lower vertebrates. For example, nothing has been built as yet that even remotely approaches the functioning of the cerebellum of even the lower amphibians and reptiles. AI is still a computer-oriented discipline at this point, but in the area of computer science

***All people in AI
agree that
we do not know
how to program
very well.***

some substantial headway is being made.

Real-World AI

Many of the large AI tool vendors have announced commercial applications that had been developed with their products this year. Companies such as Corning Glass, Campbell Soup, and American Express all have interesting expert system applications that are either completed or in stages nearing completion. American Express has developed The Authorizer's Assistant, a system that provides an automated authorization service for merchants who accept American Express charge cards. Another interesting real-word application of the technology is the Expert Publishing System developed by Crossfield CSI. This firm provides a system that helps newspaper staffs control, integrate, and coordinate the activities of various departments that have to cooperate to bring out the daily paper. The system is intended to be capable of fully integrating the layout of newspaper pages and tying together the functions of the editorial, advertising, and production departments.

Another interesting commercial ex-

pert system is the SEATS system developed by Sperry for Northwest Airlines. SEATS addresses the problem of handling discount prices in such a way as to sell as many seats on scheduled flights as possible. It acts as an intermediate operator that interacts with two other software systems—the Airline Reservation System and the Airline Revenue Enhancement System.

Several firms specialize in selling finished expert system applications for the large business market, too. For example, Applied Expert Systems is currently selling a generic business expert system, called PlanPower, that provides assistance in financial planning. The system is capable of analyzing more than 125 different types of financial asset, such as securities, insurance, real estate, and various types of investments as well as providing a comprehensive plan for a five-year period that integrates various recommendations for a specific client or user.

PlanPower is sold as a turnkey system, packaged with a Xerox 1186 AI workstation and an HP Laserjet printer. One important advantage of the Xerox workstation is that it includes an interface that allows it to run software written for IBM PC series computers. The price for all this is about \$50,000. PlanPower is supposed to be able to recommend various financial strategies based upon people's attitudes and objectives as well as their financial circumstances. Another company selling generic business expert systems is Palladian, which offers the Capital Investment Expert System and the Manufacturing and Logistics Expert System.

It is not true by any means that all serious expert system development is being done on expensive LISP machines with \$50,000 development tools. A surprising amount of substantial AI has been done already on IBM PCs, PC/ATs, and Apple Macintoshes. ARCO, for example, has developed the Cementing Expert System with the M.1 tool from Teknowledge running on an IBM PC/AT. Another oil company that has developed an expert system using M.1 is Phillips Petroleum. Other firms that have developed expert systems using the M.1 tool include Gould Electronics and the First National Bank of Chicago.

The expert system tool that captured the knowledge of retiring Aldo Cimino at Campbell Soup was the Personal Consultant tool developed by Texas Instruments for PCs and compatibles.

Although there was no particular presence of it at either the 1985 IJCAI or the AAAI conference in 1986, it would be impossible to discuss the state of the art in AI without some mention of the CYC project that is being undertaken by Doug Lenat at MCC in Austin, Texas. Dr. Lenat is best known for his work on the Eurisko Discovery System program developed while he was at Stanford and Xerox PARC.

The CYC system is one of the most ambitious, long-range AI projects ever attempted. Over the next ten years, it aspires to develop a knowledge system of truly encyclopedic size that has a knowledge base of common sense knowledge as well. Although its breadth and depth are admittedly of epic proportions, in other respects the project is not as revolutionary as it might seem because it is based largely on today's AI technology rather than the AI technology of ten years from now.

What Lenat proposes to do is literally to use encyclopedias as a knowledge source and to build a deep frame-based system that not only encodes the knowledge presented in encyclopedia articles but also the common sense implied in them. The latter, in particular, is the most ambitious aspect of the project. Common sense, oddly enough, is the area that researchers still have tremendous difficulty in making consciously explicit. It is, for example, not at all clear that a frame-based conceptual hierarchy will be a powerful enough tool to model common sense. Done properly, a large conceptual hierarchy is certainly a powerful tool. But it is powerful for providing the structure of knowledge rather than the active processes that are at work when we use know-how to solve problems.

The CYC system is representative of the same trend you see at work in Stefik's approach at the Xerox PARC Intelligent Systems Group: the refocusing of AI goals from intelligence as an artifact to that of dynamic machine knowledge systems. And it is also represented in the "knowledge

is power" slogan made popular by Edward Feigenbaum. It is now no longer intelligence but executable knowledge that is power, it seems. Nevertheless, in the often mundane world of knowledge engineering, some intelligent new ways of handling knowledge are emerging.

Starplan II

I want to devote some space here to discussing the new Starplan II architecture, an expert system for satellite diagnosis and repair under development at Ford Aerospace—how it differs from the Starplan I configuration and why such drastic changes had to be made in its earlier design. Starplan I was already an interesting expert system design because of its use of a construct the developers, Ron Siemens and Jay Ferguson, called guardians. This was a conscious attempt by them to apply Marvin Minsky's "society of experts" idea. For example, the Starplan I system had monitor and metamonitor experts, among others, that each operated as independent knowledge sources in the system. Another similar construct was the alarm demons, sleeping processes that awoke as each guardian became initialized and attached themselves to appropriate values in a telemetry database.

In order to perform its task properly, an expert system such as Starplan has to carry out several interrelated functions, including monitoring, situation assessment, diagnosis, goal determination, and real-time planning. The Starplan I system consisted of five main components: guardians, monitors, metamonitor, a simulator, and a relational database.

In Starplan II this architecture underwent a drastic revision. The developers decided that each of the main functions enumerated above would have to be implemented as entirely separate functions. In Starplan I, they were all incorporated to some degree in the role of the monitors and metamonitor, which led to considerable overlap and redundancy. The five components of the new system were organized entirely according to function. They consisted of an active database, situation assessment, causal diagnosis, goal determination, and planning and command. The knowledge base became completely

unified with each of the five modules operating on it in the same shared memory. The knowledge representation was done with the utmost completeness. Every object that could be reasoned about was represented. And each of the objects represented was defined with three components: the object's own attributes, its relation to the other objects in the satellite, and a "behavioral" description of the object.

Starplan II's developers made extraordinary claims for the resulting system. They developed a hybrid knowledge representation system in which they set out to incorporate the strong points of each of the major knowledge representation paradigms and to eliminate their weaknesses by overriding them. In their presentation at the AAAI conference last summer, they claimed to have succeeded in this ambitious objective.

ATRANS

At the Société Générale Bank in Brussels, an important AI application called the Automatic Funds Transfer Telex Reader (ATRANS) is currently undergoing its final testing phase. As the name suggests, it is a program for reading telex messages in natural language and automatically translating them into the machine-readable format of the bank's automatic payment system. Developed by Cognitive Systems of New Haven, Connecticut, ATRANS uses a method of knowledge-based parsing and text analysis. The main difficulty associated with this application is the variety of expressions that are used during actual routine telex transactions. Because the basic content of most messages is predictable, however, knowledge of the nature of the types of transactions can be used as a basis for routines that can handle various expressions that might otherwise confuse a computer.

The average monetary transfer message might contain a mixture of irrelevant text, misspellings, non-standard names for various banks, and various unusual phrases and ambiguous references. ATRANS uses a large complex script about how money transfers are made between international banks to guide its analysis of the text of incoming messages. What is unique about ATRANS is the minute detail with which it analyses these

telex messages. Unlike most other programs of its type, it carefully analyzes each and every word and produces a highly complex, detailed encoding of its content. And it does this surprisingly fast for such a large LISP application. Running on a VAX 11/785 under the VMS operating system, ATRANS processes an average funds transfer telex in less than 20 seconds.

PRIDE

I also find it encouraging that several expert systems are now being developed in the area of mechanical engineering design. In the past, nearly all the design systems attempted were intended for VLSI and other types of circuit design. One of the most interesting mechanical design programs is the PRIDE system, developed at Xerox PARC. The system is intended for the design of paper transport devices inside copy machines. Although this may not sound all that earthshaking, it is a difficult achievement to develop

software that can design anything properly. It is just these tedious and routine sorts of devices that you want machines to design, though, to leave human designers more time to work on creative and challenging problems.

PRIDE is interesting too because of the general framework and basic approach it uses to accomplish its goal successfully. This framework is deliberately configured to handle a whole class of design tasks—namely, those that can be characterized as having a well-defined search space. Given this condition, the assumption is that the design process can be reduced to the task of searching the space of possible designs. The search technique used, of course, is a special one uniquely suited to this type of problem. Detailed knowledge is used to configure the search space by the creation of partial designs according to various design constraints, and further knowledge is employed to reconfigure the design when it turns out that constraints have been violated.

Each of the different types of

knowledge used in PRIDE are organized into design plans. The main problem solver that executes these design plans has the ability to search the entire design space if necessary. The basic AI technique used to implement the problem solver is a dependency-directed backtracking mechanism equipped with an advice mechanism that allows information about why a design failed to be used to select a likely direction in which to backtrack.

So in all, the framework utilized to build the PRIDE system exploits four main types of knowledge: ordering knowledge that defines the dimensions of the design space, knowledge that guides the choices along each dimension, constraints on design parameters, and modification advice for aiding redesign. These types of knowledge have been skillfully integrated into knowledge structures that operate as usable plans. To enable this to work, plans are organized in terms of goals for making specific decisions about design parameters. In the PRIDE paper handler design system, these goals include things such as the design of paper path, driver roll, and driver width. As you might expect, each of these design goals is responsible for a certain set of design variables that correspond to its task.

Each of the design goals in the PRIDE system also has various design methods assigned to it that define alternative ways in which decisions about design parameters can be made. Some examples of design methods are generators, which actually specify sets or ranges of design parameters; calculations, which apply math operators to previously determined design variables; and subplans, which specify subgoals that are needed to satisfy higher level goals.

The PRIDE system was developed as a collaborative effort between Xerox PARC and the Xerox Reprographics Business Group. A prototype version was field-tested for more than a year, during which time it was tested on actual design problems in various ongoing copier projects at Xerox. According to the evaluations made of the prototype version, it was able both to develop new designs successfully and evaluate the shortcomings of designs produced by engineers. Research is now continuing in order

FOR THE PROfessional grammer

Find out why the PROs use HIGH SCREEN™, the professional User-Interface Development Tool that goes far beyond simple screen generators like Screen Sculptor™ or Saywhat?!™ (Trade-up available).

HIGH SCREEN™ includes:

- Complete, powerful and super fast screen editor.
- Automatic field checking: variable type, range . . . Field by field or full screen option.
- Help and window management (up to 26 pop-up windows per screen). For batch files also.
- Easy Pull-down menu management.
- Extra goodies for the PRO:
 - on-line extended character set
 - scrolling within a zone
 - ability to call a screen for reference from DOS or any language editor
 - library feature to gather screens
 - Royalty free. Not copy protected.

Language independent screens: the same copy of HIGH SCREEN™ works with Basic(s), Pascal(s), C, Cobol, Fortran, dBase, Assembler, etc.

For IBM PC/XT/AT and true compatibles. DOS 2.00+. 256K RAM.

Softway, Inc.

500 Sutter St., Suite 222
San Francisco, CA 94102
(415) 397-4666

HIGH SCREEN™ is \$129 (CA res. add tax)
S&H USA \$5, CND \$10
Visa, M/C welcome. (risk-free 30 day trial)

Trademarks / Owner: Screen Sculptor / The Software Bottling Company; Saywhat?! / The Research Group; IBM PC/XT/AT / IBM Corporation; dBase / Ashton-Tate.

to improve the advice mechanism to handle those difficult situations in which many constraints fail simultaneously and in which conflicts between different advice options have to be resolved.

New Developments in AI Research

Agora

One of the more ambitious AI research projects underway is the work being done on the Agora system at Carnegie-Mellon University. Agora is really a general-purpose AI development environment that has been designed specifically to support applications written in multiple languages and those that support highly parallel problem-solving approaches. The way Agora goes about doing this is by providing a virtual machine that is independent of any particular programming model or language and that can be mapped onto a variety of different computer architectures. The current system is actually the result of two different designs and implementations that were made in 1985. One is being used for a prototype speech-recognition system running on a network of MicroVAXes and Perqs and will be extended this year to support a shared-memory parallel computer as well as additional Sun and IBM RT/PC workstations.

Agora was used to develop the ANGEL speech-recognition system, which currently consists of more than 100,000 lines of C and Common LISP code running at a speed of about 1,000 MIPS. ANGEL has been developed by a team of more than 15 programmers, and because of this large number of programmers, several different computation styles have been used to implement the complete system.

Agora has a multilayered architecture that makes particular use of the blackboard approach to knowledge-based processing. On the first layer is the heterogeneous mix of distributed hardware mentioned earlier. On the next layer is the MACH operating system, a Unix-compatible multiprocessor OS that uses the techniques of message passing, shared memory, and threads to provide the basic software that addresses the different types of hardware. The next layer above MACH is the parallel virtual machine layer, which forms the as-

sembly-language level of the Agora machine. At this level computations are programmed in C and Common LISP that use basic Agora primitives to compile higher-level routines and assign them to the various processors. Above this is the framework layer, which is the level at which application programmers work. Finally, there is the layer of the actual knowledge source clusters.

Detailed simulations of the Agora virtual machine have been conducted and the speedup factors determined for its use on a variety of different hardware types, including custom VLSI multiprocessor chips. Initial reports indicate favorable performance on a broad mix of hardware. The Agora architecture is an important step forward in designing hardware-independent AI systems for distributed and parallel processing and is bound to be the coming trend in large systems, such as those used aboard the NASA space station.

BACAS

Researchers at the University of Es-

sex in the U.K. are developing a new parallel, content-addressable memory computer architecture called the Binary and Continuous Activation System (BACAS). This system is intended specifically for storage and retrieval of knowledge structures that turn out to be particularly useful for natural language understanding. Currently, BACAS is a two-layered system with a total of 10 different types of main knowledge structures, or K-structures, and 46 types of smaller knowledge structures, or TKUs.

In some respects this system is rather simple compared with many schemes for natural language processing and content-addressable memory, but in others it is rather subtle. It has only two "levels." On one level are "micro-units," the elementary actions or indivisible items of knowledge out of which larger knowledge structures are built. On the upper level are the threshold knowledge units (TKUs), which are built from the micro-units. The TKUs are like individual scenes in a movie that can be constructed into larger

IT'S ABOUT
TIME

APL. 68000

- Macintosh
- Atari ST
- Amiga

APL. 68000 is a highly optimized 68000 Assembler based APL Interpreter which takes full advantage of these computers' special features including user-defined pull-down menus and Dialog and Alert boxes. All this, along with a complete interface to graphics, are the reasons that APL. 68000 sets the industry standard for performance and capabilities.

Also available on PC & compatibles using 10MHZ 1MB MC68000 Coprocessor for \$995. Call for details

\$295

Order direct for \$295 + shipping (\$7 US, \$10 Canada).
VISA/MC/AMEX add 4%. Check, MO or COD. Demo
Disk available for \$15 + shipping (\$2.50 US, \$6 Canada)
May be applied to full version purchase.

30 DAY MONEY BACK GUARANTEE

SPENCER ORGANIZATION
INC.

P.O. Box 248 Westwood, N.J. 07675
(201) 666-6011



Circle no. 168 on reader service card.

scripts. It is these larger structures built from TKUs that form the ten types of K-structure.

The TKUs are a kind of hybrid between the older logic threshold unit idea of Minsky and Papert in their work on perceptrons and the newer neural net pattern completion approach developed by John J. Hopfield, which is currently so fashionable. The main problem that BACAS is designed to solve is to build a system that can make easy and rapid transi-

tions between successive knowledge structures in the course of single language interpretation tasks. Earlier, the researchers had attempted to use the Boltzmann machine approach but found it unworkable for enabling such transitions easily.

At the time the BACAS system was reported, it still lacked many of the facilities that it will ultimately need to function effectively, such as representation of causality and mechanisms for temporally ordering units and role binding. As work continues on this system, it will be interesting to see how the complete system takes

shape. Continuing research is also oriented toward developing a learning mechanism for BACAS, which makes this a promising project to track. It is another example of a marriage between state-of-the-art software and hardware concepts that has considerable potential.

JUDGE

JUDGE is an another interesting program, mainly from a computer science vantage point because a commercial version would hardly be likely to become a best-seller. It is a case-based reasoning program that assumes the role of criminal court judge. It was written at Yale University by William A. Bain and is a sample of the latest in scripts from the Schankian school.

A case-based program is one that uses previous examples of a particular activity to produce new performances of the activity to meet new circumstances. Fifty-five real cases of manslaughter and assault were read into the system, selected for their varied characteristics from the viewpoint of the program's parameters. To provide the model for the sentencing judge, various presiding judges in the Superior Court of Connecticut were interviewed. The resulting system has five stages of operation that are used to formulate a criminal's sentence: interpretation of the case; retrieval of similar cases; difference analysis of the current case and those retrieved; strategy application and modification, which adjust past sentences when necessary; and the generalization facility, which allows JUDGE to derive rules based on its findings about similar cases.

The purpose of a research system such as JUDGE is not to provide a prototype for an actual automated criminal court judge. Rather, it is to explore a type of reasoning that differs significantly from most of the types of reasoning that AI systems have so far explored. The emphasis is really on case-based reasoning and learning problems as they apply in legal situations generally, rather than the problem of criminal sentencing per se. The program, therefore, attempts to break new ground in the modeling of subjective human reasoning that is of a very different kind from the diagnostic reasoning used by doctors and

Lattice® Works

SCREEN DESIGN AID (SDA) IS NOW AVAILABLE FOR RPG II PROGRAMMERS

The Lattice Screen Design Aid (SDA) utility helps Lattice RPG II programmers create and modify display screen formats during the development and testing of application programs. Instead of coding S and D specifications for the SFGR, SDA allows you to build displays directly on your PC. When the displays on the screen are as you want them, SDA creates the SFGR source file, the screen format file for the RPG program and the skeleton RPG program for the WORKSTN file; and it can optionally print out a source listing. This product now joins Lattice Sort/Merge (LSM™) and Source Entry Utility (SEU) in supporting the Lattice RPG II compiler. \$350.00

LATTICE ANNOUNCES NEW SCIENTIFIC SUBROUTINE PACKAGE

SSP/PC is a library of mathematical subroutines essential to scientific, engineering and statistical computations. Comprised of more than 145 subroutines callable from FORTRAN, Pascal, BASIC and C, SSP/PC is as extensive as similar packages generally used on mainframe computers. The routines are very fast and extremely accurate.

and provide extensive error diagnostics. The Error Messages save the user from inadvertent mistakes. Using SSP/PC, scientists and engineers can save time by freeing themselves from tedious and difficult programming.

SSP/PC functions include:
34 Elementary Fcns, 18 Probability and Statistical Fcns, 15 Random Number Generators, Ei(x), E_n(x), li(x), Si(x), Ci(x), Γ(x), ψ(x), B(x,ω), I_n(a,b), erf x, S(x), C(x), J_n(z), Y_n(x), I_n(z), K_n(x), Ai(x), Bi(x), Ai'(x), Bi'(x), ber x, bei x, ker x, kei' x, K(x), E(x), F(ρ|a), E(ρ|a), Π(ρ|a,b), A(a,b,ρ), P(z), P'(z), P_n(x), H_n(x), L_n(x), J_n(x), G_n(p,q,x), C_n(x) and many more. \$350.00

LATTICE NOW OFFERS CODE SIFTER

Code Sifter is a software development tool that enables programmers to write faster executing software. It produces CPU usage statistics that indicate which code sections are the heavy CPU users. Using this information you can concentrate your optimization efforts on the areas that are really the bottlenecks and ignore the routines that are light CPU users.

A major advantage of Code Sifter over other products of this type is that it does not require that the user have knowledge of the machine architecture or assembly language. Link map listings are optional. In most cases Code Sifter can set up the ranges and repeatedly subdivide them automatically, freeing the programmer from a lot of drudgery. \$119.95



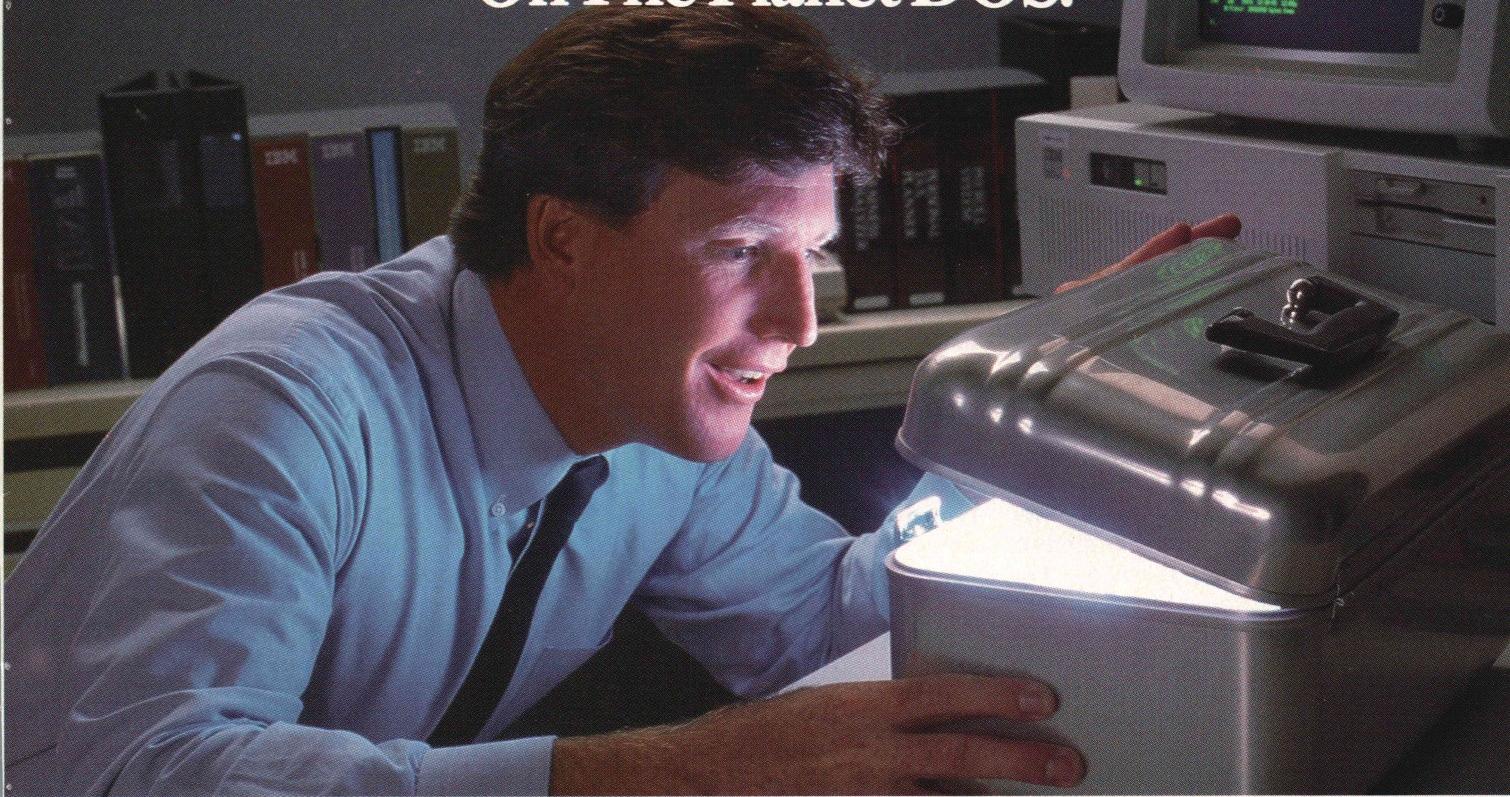
Lattice

(312)858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES: Benelux: Ines Datacom (32) 2-720-51-61
Japan: Lifeboat, Inc. (03)293-4711 England: Roundhill (0672)54675
France: SFL (1)46-66-11-55 Germany: Pfotenhaus (49)7841/5058
Hong Kong: Prima 85258442525 A.I. Soft Korea, Inc. (02)7836372

Circle no. 101 on reader service card.

Unleash The Most Powerful Development Tools On The Planet DOS.



UNIFY DBMS/DOS. The UNIX World Leader Brings A New Dimension To DOS Application Development.

What happens as the DOS world expands? As a new generation of hardware takes over? As networking becomes more important? The potential is enormous. But until now, the tools to achieve it have been limited.

Now a leader from another world unleashes that potential: UNIFY® DBMS. The leading relational DBMS in the UNIX™ world. And now, the most advanced set of application development tools in the DOS world.

With UNIFY DBMS, DOS developers have new power to build more sophisticated applications than ever before possible.

The power to write high performance "C" programs that will access the data base, using Unify's Direct Host Language Interface.

The power of an industry standard query language—SQL.

The power of unmatched speed in production applications. Only UNIFY DBMS is specifically engineered for transaction throughput. With unique performance features like PathFinder™ Architecture multiple access methods, for the fastest possible data base access.

The power of comprehensive program development and screen management tools. Plus a state-of-the-art fourth generation report-writer.

What's more, with UNIFY DBMS, the potential of networked applications becomes a reality. Unlike DBMS systems which were originally single-user (and which have a long stretch to accommodate more users), UNIFY DBMS is a proven multi-user system.

And because UNIFY DBMS/DOS is the best of two worlds, it offers you the most powerful benefit of all: DBMS applications that can grow as your needs grow. From single user DOS. To networked DOS. To multi-user UNIX. All without changing your applications.

Call the Unify Information Hotline for our free booklet: **The New DOS World.** (503) 635-7777



UNIFY
CORPORATION

4000 Kruse Way Place
Lake Oswego, OR 97034

equipment troubleshooters.

Butterfly LISP

Currently, several projects are afoot to develop new parallel hardware specifically for the rigors of AI applications. In most cases, an extended dialect of LISP is created specifically to make use of these machines. One of the most interesting of the new crop of AI supercomputers is the Butterfly Machine that has been built at Bolt, Beranek, and Neumann. Basically, the Butterfly Machine is an example of a coarse-grained parallel architecture as opposed to the massive parallelism of the Connection Machine. The Butterfly Machine uses numerous processing nodes (up to 256) each of which consists of a 68000, between 1 to 4 megabytes of RAM, and a custom processor node controller (PNC). It is from the PNC units that the Butterfly Machine gets its name. The PNCs are programmed in microcode to enable inward and outward Butterfly switch transactions and to extend the instruction set of the Motorola 68000 for multiprocessing.

Originally, all the programming on the Butterfly Machine was done in C. Based, however, on work done by Robert Halstead and the Multilisp group at MIT, an extended version of Common LISP that was also based on some features of the Scheme dialect has been implemented for the Butterfly Machine. One of the main features of Butterfly LISP is the future construct. Its form is simple. The syntax used is:

(future < expression >)

where the expression can be any LISP expression whatsoever. The future construct is used as the basic task-creating mechanism in Butterfly LISP. When the user makes a call to the system using the future construct, if resources are available, the computation is begun and control returns immediately to the function that made the call to *future*, returning a novel LISP object called an "undetermined future." This future object then acts as a temporary placeholder for the ultimate value of the expression and as such can be stored or ma-

nipulated in any fashion just as the final value would be. This, of course, is extremely significant because it means that the various computations often do not have to wait for the value of the needed expression but can continue on with their own operations as if the needed value were already available. Naturally, however, any operation that includes a conditional that depends on the value of the future expression will have to be suspended until that value becomes available. The implications of this are far reaching. It means that the results of parallel processing can be manipulated without explicit synchronization and that the form of parallel LISP programs can be essentially similar to the same programs written for sequential machines.

New AI Products

In later columns I'll be giving concrete examples of what some of the newest and most powerful AI tools can do. Here is a rather extensive preview of some of the products that you'll be reading about in far more detail in the months ahead.

ACORN

First, then, is ACORN from Gold Hill Computers. ACORN is a LISP-based expert system development tool that seems to be one of the largest and most sophisticated that has appeared so far for microcomputers. It runs on the large-model 286 Developer version of Golden Common LISP for IBM PC/ATs and compatibles. One of the reasons why I am excited about ACORN is the definite influence I see of the ideas of MIT's Dr. Carl Hewitt, the primary advocate of the actor model of programming. Gerry Barber, the chief scientist at Gold Hill was, of course, a student of Hewitt's at MIT. ACORN is not a true actor system, but it would probably be correct to say that it is the expert system tool that shows the most influence of the actor programming model.

ART 3.0

It would be wrong not to mention the new version of the ART tool, Release 3.0, even though its cost still keeps it well out of most people's reach. It is a LISP-based expert system tool from Inference Corp. that was originally sold exclusively for LISP machines

but now will be available in a C version for Sun workstations and the IBM RT/PC. One of the real delights of ART is the Artist system, an impressive graphics and animation package. ART was the first commercial expert system tool to offer the capability of multiple hypotheses reasoning using the viewpoints construct.

ExperOPS5-Plus and

ExperProlog II

Expertelligence has made two important additions to its line of products for the Macintosh. ExperOPS5-Plus is a more powerful version of the ExperOPS5 product and includes a toolkit for adding dialog boxes to provide a much-needed user interface to systems built with OPS5. Dialog boxes can be used to display bit-mapped images that have been developed with MacPaint or MacDraw. Another addition in ExperOPS5-Plus is access to the real-time speech synthesis functions provided with ExperLisp Plus, available separately. ExperProlog II is an implementation of PROLOG II, the new standard for the language proposed by Alain Colmerauer, the inventor of PROLOG. ExperProlog II takes full advantage of the Mac desktop to provide a convenient interactive development environment, as well as a facility for partitioning knowledge bases into separate contexts or worlds. ExperProlog II can be configured to use up to 4 megabytes of memory.

Expert Development Package

Arity Corp. is now selling an expert system shell, called the Expert Development Package, that runs in Arity's powerful implementation of PROLOG. The package comprises two related languages—the Taxonomy language, which offers a frame-based object hierarchy with built-in inheritance, and a rule-based language that supports an English-like syntax for rules. The system is designed for PCs and PC compatibles with 640K and support for the Above Board expanded memory standard. Initial testing of this package indicates that the expanded memory option is necessary for any serious application work. The advantages of this system are the open architecture, which allows a two-way interface between it and PROLOG, and the fact that Arity's PRO-

Subscribe to Micro/Systems Journal



**Yes! I want to subscribe to
Micro/Systems Journal
and save over 15% off the cover price.**

1 Year (6 issues) \$20 2 Years (12 issues) \$35

Please charge my: Visa MC American Express
 Payment enclosed Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Canada & Mexico add \$3 for surface mail; \$7 for airmail. All other countries add \$12 for airmail. All foreign subscriptions
must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

300



**Yes! I want to subscribe to
Micro/Systems Journal
and save over 15% off the cover price.**

1 Year (6 issues) \$20 2 Years (12 issues) \$35

Please charge my: Visa MC American Express
 Payment enclosed Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Canada & Mexico add \$3 for surface mail; \$7 for airmail. All other countries add \$12 for airmail. All foreign subscriptions
must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

300



**Yes! I want to subscribe to
Micro/Systems Journal
and save over 15% off the cover price.**

1 Year (6 issues) \$20 2 Years (12 issues) \$35

Please charge my: Visa MC American Express
 Payment enclosed Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Canada & Mexico add \$3 for surface mail; \$7 for airmail. All other countries add \$12 for airmail. All foreign subscriptions
must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

300



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

For the Advanced Computer User

Micro/Systems Journal™

501 GALVESTON DR.
REDWOOD CITY, CA 94063



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

For the Advanced Computer User

Micro/Systems Journal™

501 GALVESTON DR.
REDWOOD CITY, CA 94063



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

For the Advanced Computer User

Micro/Systems Journal™

501 GALVESTON DR.
REDWOOD CITY, CA 94063



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

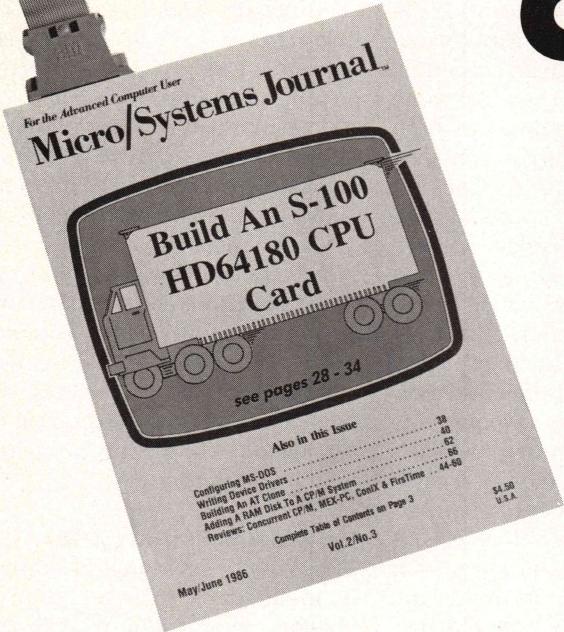


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



**Subscribe to
Micro/Systems Journal**

Your System's Key Component



For the Advanced Computer User

Micro/Systems Journal™

At last there is a magazine that brings you to strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . .

In future issues of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that Micro/Systems Journal is famous for . . . indepth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal*, simply fill out one of the attached cards or write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just \$20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!

Ever Program On A Silver Platter??

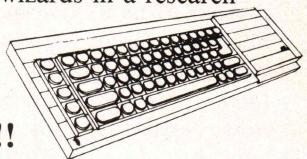
How much would you expect to pay for a 32 bit MC 68000 computer that's a mainframe condensed down into a keyboard? How about \$177.00!!! If it makes you feel any better simply add a zero to the price when you order! But that's actually our price!!! The most powerful computer money can ever buy is now the most inexpensive computer money can buy!!! So don't buy the name! Buy the power!! The power is **not** in the name!

If you had the opportunity to work amongst Machine Code ROM Designers, VAX & UNIX wizards in a research laboratory, designing an MC 68000 based computer that's 2nd to none. . .

What would you come up with?? And what would you call it??

Well It's Already Been Done!!

They Called It The QL For The Quantum Leap It Is!!



Absolutely a Quantum Leap beyond what you know & use - and it's truly like Programming on a Silver Platter!!

The QL Desktop Minicomputer: Designed by SRL Labs, manufactured by Samsung. A total Quantum Leap beyond all the rest! Virtual Memory, Multitasking Job Control, Multiuser Networking, Drives in Cache Memory, with Automatic Directories & File Names up to 36 characters! Everything is built into ROM! QDOS, Networking, 32 Bit SuperBasic: a totally concurrent non-destructive environment. Unlimited quantities & lengths of: Variables, Program Lines, CONsoles & Buffers. Dynamic RAM Disking! Even a System Variables Brain Page Screen! Built-in DCE & DTE Serial Ports.

Imagine a 32 bit SuperBasic structured like Turbo Pascal, always there with QDOS in a UNIX-like multitasking job controlled environment with access to 32768 fully channelled windows, devices & files by EACH job! 3 Major Compilers exist for SuperBasic source alone! TURBO, SUPERCHARGE, LIBERATOR. Also runs "C", Pro Pascal, Pro Fortran 77, LISP, BCPL, 68000 Assemblers, JCL, APL, Forth-83, 65C02 or 8088 Cross Assembly. Generate native 68000 code. Compile SuperBasic source code or ANY other language and then multitask and control it in QDOS or even with SuperBasic in the Interpreter! The list of ALL the Superior Features would fill this entire journal!

Assembled QL comes WITH Integrated WP, SS, DB & Presentation Graphics Program Package by PSION! Also available in a built, final assembly KIT Form for another \$25.00 LESS!!! PLUS: FREEWARE Demos & Utilities with all purchases!

Call: (201) 328-8846

QLine BBS: 328-2919

Quantum Computing, Box 1280, Dover, NJ 07801

Technical Info & Assistance --

Our Phones are Manned 24 Hours a Day
Lot Purchases Available. We Direct Distribute.



ARTIFICIAL INTELLIGENCE

(continued from page 116)

LOG compiler can be used to develop stand-alone expert systems in standard .EXE machine-code format.

The Intelligence/Compiler

The Intelligence/Compiler from IntelligenceWare is another exciting expert system shell that is helping to make all those people who said that professional AI systems cannot be supported on microcomputers think again. This system supports four different programming paradigms: rule-based programming, frame-based representation, logic-based programming, and procedural programming. Other pluses with The Intelligence/Compiler include a smart editor and a built-in B-tree database manager. The editor includes its own built-in parser that can detect syntax errors, and word completion and spelling checking are also planned.

KEE 3.0

Intelicorp has taken the first step toward updating its KEE system to keep pace with the competition. In Release 3.0 it has introduced the KEEworlds facility, which is intended to provide KEE with the capability of hypothetical reasoning with multiple hypothetical continuations. The KEEworlds facility is based on the ATMS (assumption-based truth maintenance) approach to truth maintenance developed by Johann deKleer at Xerox PARC. Full integration with all other facilities of KEE is provided with this new capability, and an interactive user interface featuring a World-Browser has also been provided.

Nexpert Object

The only way to classify Neuron Data's new Nexpert Object system for IBM PC/ATs and compatibles—a significantly more powerful version of the system available for the Mac—is as an extremely imaginative product that is very much in a category all its own. This is easily the tool that comes the closest of any so far to creating an AI workstation environment on a micro. In addition to the lush user interface running under Microsoft Windows, Nexpert Object has several powerful and unique features. One of my favorites is the common rule

format for both forward and backward chaining. As far as I know, this is the only commercial expert system shell that offers this feature.

Most expert system shells that offer fully specifiable forward and backward chaining use a different format for forward and backward chaining rules, which means that the same knowledge cannot be used in both reasoning modes. With a common format for both types, the opportunity exists for writing "pure knowledge" rules that are usable for a variety of purposes and can be accessed in either type of inference.

The main new addition to the system is a major one. Nexpert Object has a true object-oriented class hierarchy system whereas Nexpert for the Mac does not. An object editor is provided to make this facility accessible to users. Each object has properties (or slots) as well as subobjects. A subobject is a part of the object that constitutes an object in its own right, such as an organ that is part of a person's body. An object also has metaslots, which provide various ways for the object to utilize various custom features regarding such things as its salience, inheritance relations, or sources of information.

Office Automation Toolkit

One interesting tool available for LISP programmers is the Office Automation Toolkit from Grandmaster. This is a high-level library of programming tools that facilitate the development of business-oriented applications in LISP. This package is built for the muLISP-86 dialect available from Soft Warehouse and Microsoft. Functions provided in this toolkit allow programmers to develop applications that use multiple buffers, multiple windows, menus, forms, and natural language commands. A new addition to the toolkit also provides functions for programming applications that utilize the B+-tree database structure. One of the attractions of this toolkit is that, like libraries written for compiled languages, no royalties have to be paid on stand-alone applications that use it.

PC Scheme 2.0

Difficult as it may be to choose, if pressed, I would have to say that the most remarkable implementation of

LISP, as well as my favorite object-oriented programming environment for PCs so far, is PC Scheme from Texas Instruments. PC Scheme is the most modern and streamlined of the LISP dialects and is the one many think should have been the basis for Common LISP. PC Scheme provides an object-oriented extension called SCOOps that features the Mixin multiple-inheritance capability and active values, formerly found mainly on expensive LISP machines. PC Scheme includes a compiler and a smart, Emacs-style, full-screen editor. At less than \$100, this is easily the best buy in LISP systems for PCs. If it were not for TI's extreme low profile in marketing this product, it probably would have already become the LISP counterpart to Turbo Pascal. It undoubtedly will in any case.

Personal Consultant Plus

TI has also rewritten and significantly updated its expert system shell product, which now runs in the PC Scheme environment. Personal Consultant Plus offers four main features that are of particular interest: frame-based representation, metarules, active value access methods, and an open architecture. The open architecture in this case is particularly interesting because it means that it is possible to construct some advanced "deep" systems with Personal Consultant Plus by exploiting SCOOps, the object-oriented extension to PC Scheme. Other useful features are the ability to display graphic images and a Snapshot utility that can import graphics developed using any other software package.

VP-Expert

Paperback Software's VP-Expert program is likely to become the same kind of spoiler in the commercial AI software category that Turbo Pascal was in the structured programming segment of the market. It has just too many features to be sold for less than \$100—or so all its competitors will undoubtedly feel. The irrepressible Adam Osborne obviously has different thoughts on the subject. First impressions of it are that it is something like a poor man's Guru or a poor man's M.I. But this is not quite right either because VP-Expert has other features that make it comparable in

WE JUST GOT MORE SOPHISTICATED SO YOU CAN GET MORE BASIC.

We invented BASIC over 20 years ago.

Later, we re-invented it for micros as the True BASIC™ structured-programming language. And the idea was: To make programming as easy and natural as possible. So you could concentrate on what to program. Not how.

Now there's True BASIC Version 2.0 for the IBM® PC and compatibles. Faster, more powerful and sophisticated than the original.

MORE GRAPHICS.

Right from the start, True Basic gave you terrific device-independent graphics. Built-in 2-D transforms. And support for multiple windows.

Now we've added more graphics and full mouse support.

So for the first time, you can create one program that will do superb graphics on CGA, EGA or Hercules displays. Without worrying about additional drivers or overlays. And on the EGA, you can SET COLOR MIX to define your own colors. Use four shades of blue if you want (and make our competitors green with envy).

MORE CONTROL.

We always supported you with recursion, local and global variables and separately compiled libraries.

Now you can have *modules*, too, the industrial-strength tool for building large applications.

Using modules makes it easier for you to share data between routines. Build data structures. Then, if you want, hide them from other parts of the program. So you can always be free to focus on the task at-hand.

Modules have their own initialization sections, so you can set up global variables or turn on instrumentation.

And, like other procedures in True

BASIC, modules can be compiled separately and stored in a library where they can be shared by several applications. Or they can be loaded directly into the True BASIC environment as part of your customized workspace. So when you use True BASIC interactively, the modules look like built-in functions.

Modules made Modula-2 the successor to Pascal. Now they've put True BASIC one-up on all other BASICS.

MORE SPEED.

2.0 is 20 to 200 percent faster than True BASIC Version 1.0. Both compile times and execution speeds. And on some real-world benchmarks, we're faster than many native-code compilers.

MORE POWER.

Start with a complete matrix algebra package.

Then, since we support the use of 640K for both code and data, add arrays as large as you want.

Our compiled code is more compact than what other compilers generate, so there's more memory left for your application.

We've enhanced our dynamic array redimensioning and improved our built-in 8087/80287 support, making True BASIC the most powerful number-crunching BASIC around.

And if it's strings you crunch, we've added new string functions and raised the limit. So strings can be up to 64K characters long.

MORE DEBUGGING.

We pioneered breakpoints and immediate-mode capability in a compiled BASIC environment.

Now we've added utilities that allow you to visually TRACE through your program, and check the values of selected variables. Or print a cross-referenced listing.

And new compiler options like NO LET and NO TYPO let you decide how strictly you want your variable names checked.

MORE INNOVATION.

True BASIC has always had features like full-screen, scrollable editing. Block copy and block moves. And global search and replace.

Now, 2.0 keeps you on the leading edge of editing and file-management technology. With SCRIPT, to write the True BASIC equivalent of a DOS batch file. ECHO, to transfer your output to disk or printer. And ALIAS, to give you and your programs a better roadmap to your subdirectories.

There's also Version 2.0 of the Developer's Toolkit. With support for DOS interrupts. Pop-up menus. Even designer fonts.

And remember: your programs are portable to the other machines we support: the Apple Macintosh™ and Commodore Amiga®.

MORE SUPPORT.

Call your local dealer. Call us TOLL-FREE at 1-800-TR-BASIC. Or write to: True BASIC, Inc., 39 South Main Street, Hanover, NH 03755. We'll send you more information. Including a free demo disk.

See for yourself. That we're still true to our basic idea.

True
BASIC™
inc.

True BASIC Language System is a trademark of True Basic, Inc. Macintosh is a trademark licensed to Apple Computer Inc. Amiga is a registered trademark of Commodore-Amiga, Inc. IBM is a registered trademark of International Business Machines.

ARTIFICIAL INTELLIGENCE
(continued from page 118)

some ways to programs such as Expert Ease. Suffice it to say that it is now easily the runaway winner for the "best buy" award in PC expert system software.

New AI Books

Right now, with the enormous changes in AI—both those already occurring and the ones still ahead—books are playing as much or more of

a role in leading the way as are important AI programs and applications. Like it or not, books reach a lot more people than great programs do. Here are some new ones that look especially interesting:

Agha, Gul. *ACTORS*. Cambridge, Mass.: MIT Press, 1986.

Galambos, James A., et al. *Knowledge Structures*. Hillsdale, N.J.: Lawrence Erlbaum Assoc., 1986.

Schank, Roger. *Explanation Patterns*. Hillsdale, N.J.: Lawrence Erlbaum Assoc., 1986.

Sterling, Leon, and Shapiro, Ehud. *The Art of Prolog*. Cambridge, Mass.: MIT Press, 1986.

Wilensky, Robert. *Common LISPcraft*. New York: Norton, 1986.

Winograd, T., and Flores, F. *Understanding Computers and Cognition*. Norwood, N.J.: Ablex Publishing Corp., 1986.

Bibliography

Bain, W. M. "A Case-Based Reasoning System for Subjective Assessment." *AAAI-86*.

Bisiani, R. "A Software and Hardware Environment for Developing AI Applications on Parallel Processors." *AAAI-86*.

Blelloch, G.E. "CIS: A Massively Concurrent Rule-Based System." *AAAI-86*.

Brooks, R., et al. "A Mobile Robot with Onboard Parallel Processor and Large Workspace Arm." *AAAI-86*.

Edelson, T. "Can a System Be Intelligent if It Never Gives a Damn?" *AAAI-86*.

Hammond, K. "CHEF: A Model of Case-Based Planning." *AAAI-86*.

Hon Wai Chun. "A Representation for Temporal Sequence and Duration in Massively Parallel Networks." *AAAI-86*.

Lenat, D., et al. "CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks." *AI Magazine* (Winter 1986).

Lozano-Perez, T. "A Simple Motion Planning Algorithm for General Robot Manipulators." *AAAI-86*.

Lytinen, S., and Gershman, A. "ATRANS: Automatic Processing of Money Transfer Messages." *AAAI-86*.

Sharkey, N., et al. "Mixing Binary and Continuous Connection Schemes for Knowledge Access." *AAAI-86*.

Siemens, R., et al. "Starplan II: Evolution of an Expert System." *AAAI-86*.

Stefik, M. "The Next Knowledge Medium." *AI Magazine* (Spring 1986).

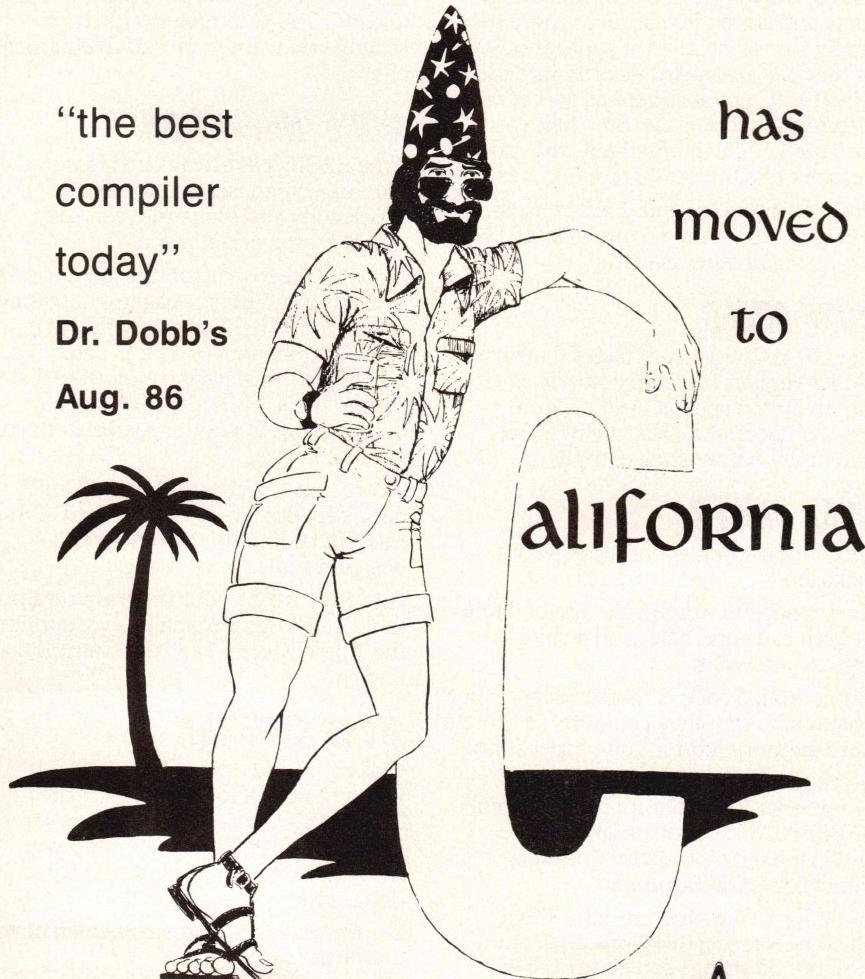
Steinberg, S., et al. "The Butterfly LISP System." *AAAI-86*.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 7.

WIZARD systems

"the best
compiler
today"
Dr. Dobb's
Aug. 86



(408) 224-6015

Only \$450



Circle no. 116 on reader service card.

WIZARD
SYSTEMS SOFTWARE, INC.

17645 Via Sereno
Monte Sereno, CA 95030

QUIT DOING GRUNT WORK.

Let Greenleaf do it for you and set you free.

C Program developers, stop slaving!

Greenleaf libraries have the functions you need — already perfected and in use by winning program developers in major corporations such as IBM, EDS and GM.

Between our Greenleaf Functions and Greenleaf Comm Library, we have over 340 functions on the shelf. Each one can save you time and effort. Money, too.

Many C programmers have told us that, even if they only use one or two functions, our products easily pay for themselves:

The Greenleaf Functions

The most complete and mature C language function library for the IBM PC, XT, AT and close compatibles. Our version 3.0 includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, new disk status and Ctrl-Break control functions plus many more!

The Greenleaf Comm Library

Our 2.0 version is the hottest communications facility of its kind. Over 120 all new functions — ring buffered, interrupt-driven asynchronous communications.

Call Toll Free

1-800-523-9830

In Texas and Alaska, call

214-446-8641



GREENLEAF
Software

Greenleaf Software, Inc.
1411 LeMay Drive Suite 101
Carrollton, TX 75007

If you need more than 2 ports, only Greenleaf gives you the total solution — boards, software, and complete instructions that enable you to build a 16-port communication system.

And no matter how many ports you have, it's virtually impossible to lose information with multiple file transfers. XMODEM, XON/XOFF and Hayes modem controls are featured.

We support all popular C compilers for MS DOS: Lattice, Microsoft, Computer Innovations, Wizard, Aztec, DeSmet and Mark Williams.

Order today!

Order a Greenleaf C library now. See your dealer or call 1-800-523-9830. Specify compiler when ordering. Add \$8 for UPS second day air, or \$5 for ground. Texas residents, add sales tax. Mastercard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf

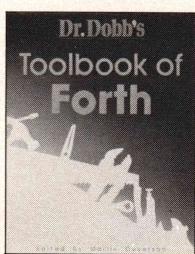
| | |
|--------------------------|-------|
| Comm Library v2.0 | \$185 |
| Greenleaf Functions v3.0 | \$185 |
| Digiboard Comm/4-II | \$315 |
| Digiboard Comm/8-II | \$515 |

We also sell compilers, books and combination packages.

Dr. Dobb's Toolbook Shelf

Dr. Dobb's Toolbook of Forth

Ed. by Marlin Ouverson

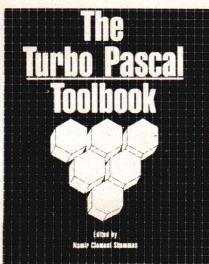


This comprehensive collection of useful Forth programs and tutorials contains Dr. Dobb's best Forth articles, expanded and revised, along with new material. Contents include: Mathematics in Forth, Modifications/Extensions, Forth Programs, Forth--The Language, Implementing Forth. All screens in the book are also available on disk as ASCII files.

Toolbook of Forth #030 \$22.95
Toolbook of Forth with disk

Item #031 \$39.95

Please specify one of the following disk formats: MS/PC-DOS, Apple II, Macintosh, CP/M. For CP/M disks, specify Osborne or 8" SS/SD.



The Turbo Pascal Toolbook

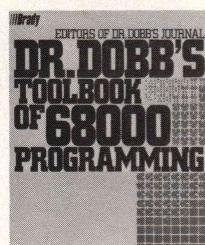
Ed. by Namir Clement Shamas

This book contains routines and sample programs to make your programming easier and more powerful. You'll find: an extensive library of low-level routines; external sorting and searching tools; window management; artificial intelligence techniques; mathematical expression parsers including two routines that convert mathematical expressions into RPN tokens; and a smart statistical regression model finder. All routine libraries

and sample programs are available on disk for MS-DOS systems. Over 400K of Turbo Pascal source code is included.

Turbo Pascal Toolbook Item #080 \$25.95
Turbo Pascal Toolbook with disk Item #081 \$45.95

Dr. Dobb's Toolbook of 68000 Programming



In this complete collection of practical programming tips and techniques for the 68000 family, you'll find the best articles on 68000 programming ever published in Dr. Dobb's, along with new material from 68000 experts. You'll find a set of development tools, routines, useful applications, and examples to show you why computers using the 68000 chip family are easy to design, produce and upgrade. All programs are available on disk.

Please specify one of the following disk formats: MS-DOS, CP/M 8", Osborne, Macintosh, Amiga, Atari 520st.

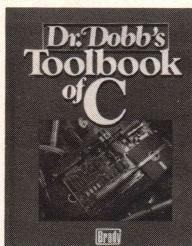
68000 Cross Assembler

An executable version of the 68000 Cross-Assembler discussed in the book is also available, complete with source code and documentation. Requires CP/M 2.2 with 64K or MS-DOS with 128K. Specify: 8" SS/SD, Osborne, MS-DOS

68000 Toolbook
68000 Toolbook with disk
68000 Cross Assembler

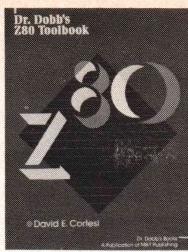
Item #040 \$29.95
Item #041 \$49.95
Item #042 \$25.00

Dr. Dobb's Toolbook of C



This authoritative reference contains over 700 pages, including the best C articles from Dr. Dobb's Journal, along with new material. You'll find hundreds of pages of valuable C source code, including a complete compiler, an assembler, and text processing programs.

Toolbook of C Item #005 \$29.95



Dr. Dobb's Z80 Toolbook

by David E. Cortesi

This book, along with its companion disk, contain everything you need to write your own Z80 assembly language programs. You'll find a method of designing programs and coding them in assembly language, and a complete, integrated toolkit of subroutines for arithmetic calculations, string handling and total control of the CP/M file system. Every line of the toolkit's source code is there to read. All the software, both as source code and

object modules for CP/M 2.2 and CP/M Plus, is available on disk. (Most programs are included in the book; however, the disk is necessary for complete listings. A Z80 microprocessor and a Digital Research International RMAC assembler or equivalent are required.)

Z80 Toolbook

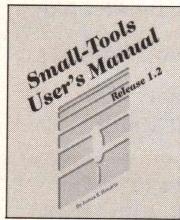
Item #022 \$25

Z80 Toolbook with disk

Item #022A \$40

Please specify one of the following disk formats: 8" SS/SD, Apple, Osborne, Kaypro.

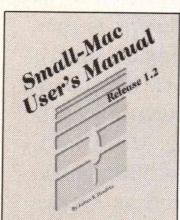
Small-Tools: Programs for Text Processing



This package of Small-C programs performs specific, modular operations on text files. It is supplied as source code with full documentation. With the Small-C Compiler, you can select and adapt these tools to meet your needs. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Tools Item #010A \$29.95

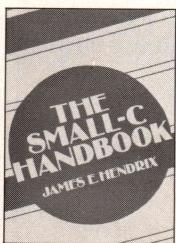
Small-Mac: An Assembler for Small-C



This package includes: a simplified macro facility, C language expression operators, object file visibility, descriptive error messages and an externally defined instruction table. Source code and documentation is included. CP/M only. Please specify: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Mac Item #012A \$29.95

The Small-C Handbook & Small-C Compiler



This compiler and handbook provide everything you need for learning how compilers are constructed, and for learning C at its most fundamental level. You'll find a discussion of assembly language concepts and program translation tools, and of how to generate a new version of the compiler. Full source code is included. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

CP/M Compiler & Handbook Item #006B \$37.90
MS/PC-DOS Compiler & Handbook Item #006C \$42.90

Order Form

To Order: Return this order form with your payment to M&T Books, 501 Galveston Dr., Redwood City, CA 94063

Or, Call **TOLL-FREE 800-533-4372** Mon-Fri 8AM-5PM
In CA call **800-356-2002**

Name _____

Address _____

City _____ State _____ Zip _____

| Item # | Description | Price |
|--------|-------------|-------|
| | | |
| | | |
| | | |
| | | |

Subtotal _____

CA residents add sales tax % _____

Add \$2.25 per item for shipping _____

TOTAL _____

Please specify disk format

MS/PC-DOS Macintosh Apple II Amiga Atari 520st
CP/M Kaypro Osborne Apple
 Zenith Z-100 DS/DD 8" SS/SD

Check enclosed. Make Payable to M&T Publishing.

Charge my VISA M/C Amer.Exp.

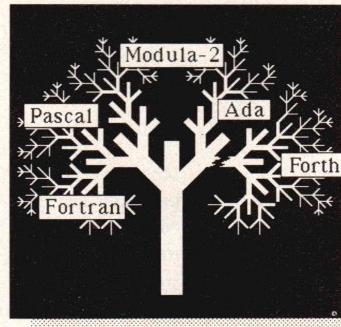
Card No. _____

Exp. Date _____

Signature _____

STRUCTURED PROGRAMMING

Language Translations



This column begins a series of discussions regarding the translation of programs between different languages or implementations. I'll start by looking at selected cases of translating Turbo Pascal programs into Logitech Modula-2.

Logitech has recently released the Translator, which allows the migration of memory-hungry Turbo Pascal programs to Modula-2/86. Discussing the Logitech Translator at length would fill a book; instead, I'll present four sample Turbo Pascal programs and discuss their translations.

The first question you might ask about the Translator is, "Does it translate 100 percent?" For most applications, the oversimplified answer is no. The Translator may include several warning messages in a generated Modula-2 listing, but the absence of such messages does not necessarily mean that the program compiles correctly. As a matter of fact, the Translator deliberately leaves some types of errors (those due to incomplete translation) for the compiler to detect.

Before you translate a program, you should first identify its components: expressions, loops, decision-making constructs, procedures, functions, and so on. The Translator can translate similar language aspects, such as loops, decision-making constructs, and routines, from Pascal to Modula-2 without the need for "manual" editing. Any difficulties in trans-

**by Namir Clement
Shammas**

lation are partially because of differences in data types and their manipulation.

Numeric Manipulation

Listing One, page 70, shows a Turbo Pascal program that implements a simple four-function calculator. The program deals with *REAL* numeric

manipulations. Listing Two, page 70, contains the translated Modula-2 code. No hand-coded patches were needed to make the Modula-2 version run correctly. Note that the *REPEAT...UNTIL* loop and the *CASE* and *IF...THEN* constructs have been translated correctly. The Modula-2 version contains more code, especially for I/O operations—for example, a single Pascal *WRITELN* statement has been replaced by several Modula-2 procedures. Notice the use of *stdinout* as the standard I/O device in the Modula-2 translation. In addition, the output procedures in Modula-2 contain arguments for formatting the displayed numbers.

A note on translating a Turbo Pascal *CASE* statement without the *ELSE* clause: in Turbo Pascal, if the value of the *CASE* variable does not match any option, program flow simply resumes outside the *CASE* construct. A similar situation in Modula-2 yields a run-time error. Thus, the Translator inserts an *ELSE* clause in the *CASE* construct if one is not present in the Pascal program.

The first example translates without any snags. Programs that tackle sorting and searching of *INTEGERS* and *REALS* fall into the same category. However, operations such as bit manipulation using *INTEGERS* and implicit numeric type conversion need additional editing. The core Modula-2 language does not support bit manipulation using *INTEGERS* or *CARDINALS*—after all, that's what the *BITSSET* type is for. This can be frustrating for Pascal programmers who take in-

teger-bit manipulation for granted. The Translator library does, however, contain procedures that support manual editing for integer-bit manipulation. For example, you can recode the following Pascal expression:

```
(Base_Address AND Offset) OR
(Mem_Loc Shl 3)
```

by hand into the following RPN-like expression in Modula-2:

```
OR(AND(BaseAddress,Offset),
Shl(MemLoc,3))
```

String Manipulation

The next example deals with string manipulations. Listings Three, page 72, and Four, page 78, contain programs that perform text pattern searches. The '?' and '*' wildcards are supported and follow the convention familiar to CP/M and MS-DOS users. String manipulation is the most common area in which a good deal of additional editing is required. First, in Turbo Pascal the *STRING* type is predefined with a lower index of 1 and a maximum length of 255. (The zero index of a Turbo Pascal *STRING* stores its maximum length.) In Modula-2, strings are handled quite differently. Basically, they are treated as arrays of characters. The current Modula-2/86 compiler requires that string types have a lower array index of 0. The reason for this seems to be a feature that enables the compiler to automatically insert an end-of-string delimiter (as in the C language) when assigning a string constant to a string variable, as in:

```
Name : ARRAY [0 .. 79] OF CHAR;
(* more definitions and statements *)
Name := 'Don Johnson';
```

Otherwise, the programmer is responsible for inserting the control 0 delimiter, as in:

```

Name := ARRAY [1 .. 80] OF CHAR;
(* more definitions and statements *)
Name := 'Don Johnson';
Name[12] := 0C;

```

Strings in Modula-2/86 can be up to 65,535 characters long, and future implementations may permit this upper limit to be around two billion (the upper range of the long integer type *LONGINT*). Moreover, the support of open arrays in Modula-2 does away with the Turbo Pascal directive '\$V-' to relax the strict string type checking with routine parameters.

Listing Four contains comments beginning with --> strings. These point out the lines emitted by the Translator that needed further editing. A few lines were inserted, and the *INC* procedure was removed because Modula-2 already has it as a predefined routine that performs the same task. The constant declaration section has a new identifier *HI*, which is used in conjunction with the substring scanning function *Pos()*. In Turbo Pascal, a value of 0 is returned if no match is found. In Modula-2, the index value of 0 points

to the first character. The implementors of the Modula-2 version *Pos()* have elected to return a value greater than the upper string dimension limit. Thus, in Turbo Pascal you test for substring match using:

```

Actor := 'Don Johnson';
(* Index is an INTEGER *)
Index := Pos('Sellek', Actor);
IF Index > 0 THEN (* statements *)

```

whereas in Modula-2 the same test is written, using the *HIGH()* function, as:

```

Actor := 'Don Johnson';
(* Index is a CARDINAL *)
Index := Pos('Sellek', Actor);
IF Index <= HIGH(Actor) THEN
    (* statements *)

```

The constant *HI* is assigned as a "consistent" large integer value that signals a substring mismatch status. It is assigned to variables that are used to monitor the first character position at which a match occurs.

You may have noticed I have used open arrays in the *PatternSearch*

function instead of the *STRING255* and *STRING40* types for declaring the two string parameters. The declaration can be rewritten as:

```

PROCEDURE PatternSearch(TextLine,
    Pattern : ARRAY OF CHAR) :
    (* function returns an *) INTEGER;

```

The body of the function has been edited to use constant *HI* and to account for the difference between the indexing schemes that Turbo Pascal and Modula-2 use.

The Modula-2 version of *PatternSearch* has a new identifier, *PatternSearchResult*, to return the function result. The Translator inserts the new identifier because in Modula-2 functions return their results via the *RETURN <identifier>* syntax and do not use the function name in an assignment statement.

In the procedure *ScanPattern* and the function *LocatePattern*, the variables tracking character positions have all been shifted by 1 and assigned an initial value of 0. Using the string-copy function *Copy()* seems to require no alteration. In the function

Changing Your Address? We'd Like to Know.

To change your address, attach your address label from the cover of the magazine to this coupon and indicate your new address below.

affix label here

Name _____

Address _____ Apt. # _____

City _____ State _____ Zip _____

Mail To:

**Dr. Dobb's Journal, P.O. Box 27809, San Diego, CA
92128**

LEARN C ON TV

**with the video training course
A Programmer's Introduction to C
by Ray Swartz**

Topics Covered: Data Types • Operators • Expressions • Loops • Conditionals • Input and Output Character Handling • Interactive Control • Arrays • Pointers • Switch

Reviewers' Comments:

"The tapes cover the major features of C and are aimed at professionals with a background in another computer language. The key to the package is that Swartz knows his subject and how to teach it." Dr. Lance Leventhal

"If you're looking for a way to learn C quickly and easily, this course is something you'll want to consider."

Doug Topham

Price: \$400 includes two video tapes (3½ hours)
and 106 page manual.

**BERKELEY DECISIONS/SYSTEMS
ASK ABOUT OUR
FREE 15 DAY REVIEW** **TO ORDER CALL
(408) 458-0500**

Circle no. 202 on reader service card.

STRUCTURED PROGRAMMING (continued from page 125)

LocatePattern, most of the tests in the *IF* statements needed editing. The original statements are still included, enclosed in comments. Also notice that the result of the string length function, which returns a *CARDINAL*, has been converted into an *INTEGER* to match the assigned variable type.

In the main program segment, the *IF Line = '' THEN* Pascal code has been rewritten as *IF Line[0] = 0C THEN* for Modula-2. In addition, string assignment between two variables needs the *Assign()* procedure instead of Pascal's := assignment operator.

Sets

The third example deals with sets. Logitech Modula-2 specifies that sets can have 16 members at most—not enough to handle the character sets that Pascal supports. The Translator emits *BITSET{}* when it encounters Pascal sets such as *{'Y', 'y'}* or *{'A'..'Z'}*. The translation needs manual editing, using a much needed LongSet module supplied with the Translator. Listing Five, page 84,

shows a simple Pascal program that performs character counting. The program prompts you for a text file; reads it; and classifies each character as either a digit, uppercase, lowercase or "others." It then draws a simple histogram (each '*' represents a hundred count) for each character category. The program is used to demonstrate handling small and large (more than 16 elements) sets.

The Translator produces a Modula-2 program (Listing Six, page 85) that requires a fair amount of editing. First, I'll discuss the small set. In the Turbo Pascal listing, a two-member set *{'Y', 'y'}* is used to examine the user's response in the *UNTIL* clause. In the Modula-2 version, I import *MakeEmpty()*, *Include()*, and *Inset()* and the *SetOfChar* type from module *LongSet*. In the Modula-2 program, I declare *YesNo* as a variable of type *SetOfChar* and use the *MakeEmpty* procedure to create an empty set of *YesNo*. The *Y* and *y* set members are inserted using the *Include* procedure. Notice the nested use of *WORD()* and *ORD()* to transform the character's ASCII code into a *WORD*. The above steps prepare the *YesNo* set for testing of

membership. The Boolean function *InSet()* is used with the ordinal value *OK*, the character-typed response.

Handling long sets is different. The *BuildSet()* procedure takes three arguments: the set-typed variable and the ordinal values of the first and last set members. Three calls are made to *BuildSet()* to create the three sets in question (I am treating *DigitSet* as a long set). The *Inset()* function is used to determine the set membership of each file character read.

Absolute Variables

The last example deals with simple absolute variables. Listings Seven and Eight, pages 86 and 87, show Pascal and Modula-2 programs that write strings directly to the screen memory of an IBM PC with a monochrome monitor. The Pascal listing contains an identifier *DISP* defined as:

```
DISP : SCREEN80 absolute $B000:0000;
```

which is edited in Modula-2 to become:

```
DISP [B000H:00H] :STRING80;
```

TURBO TOOLBOXES

FULL SCREEN I/O

This toolbox gives you as much power for keyboard, screen, and printer I/O as Borland's Database Toolbox gives you for disk I/O. It lets you read many fields from the screen in one operation.

Read a record and display all fields on the screen. Your user can tab around from field to field, changing any or all fields, before sending the whole screen back to your program.

Numeric fields act like a calculator window. String fields have insert, overwrite, and delete. Formatted fields are good for phone numbers and dates.

Use color (or intensity on monochrome) to highlight input fields. Choose a border color.

Print selected lines from the screen. No need to format a screen and then format a similar printed report.

INTEGER DATES

An integer date is the number of days since a base date.

You pick the base date. Valid dates extend 100 years into the 21st century. It's simple to compare or subtract two dates.

Converts formats like 12/31/86, 86365, 31DEC86, and DECEMBER 31, 1986. All routines check dates.

- Save time. Why write this code yourself? Make changes to suit.
- Only \$19 for Full Screen I/O, \$10 for Integer Dates, plus \$2.50 S&H per order. PA add 6%. No royalties.
- Printed User's Guide with each product.

VERTICAL HORIZONS SOFTWARE

113 Lingay Drive
Glenshaw, PA 15116

412-487-5752

Checks, Visa, Mastercard

Turbo Pascal is a trademark of Borland International Inc.

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

BTree Library

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

75.00

ISAM Driver

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

40.00

Make

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

59.00

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR — **149.00**

For more information call or write:

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

Credit cards accepted.

Dealer inquiries invited.

Circle no. 259 on reader service card.

softfocus

THE PROGRAMMER'S SHOP

Offers a 31 Day Money Back Guarantee on any product in this ad. Call Today.

COBOL PROGRAMMING PRODUCTIVITY!

screenplay™
SCREEN MANAGEMENT SYSTEM

SUBSTANTIALLY REDUCE APPLICATION DEVELOPMENT TIME!

Save valuable time by avoiding the tedious, time consuming process of writing screen handling source code. *screenplay's* easy-to-use panel painter allows you to create I/O panels, pop-up help windows or menu panels which you use in your program.

FLEXIBILITY IN PANEL PAINTING; FLEXIBILITY AT RUNTIME!

True screen handling flexibility is yours with *screenplay*, because you can override default panel settings to design practically any type of panel imaginable! *screenplay* continues to provide the flexibility you need during the operation of your program by allowing you to change just about any panel characteristic at runtime.

PROTOTYPE PANELS BEFORE YOU WRITE CODE!

With *screenplay*, you can prototype your draft screens before you write a single line of COBOL source code. These can then be reviewed with your boss or your customer for final approval, before you start writing source code.

MORE THAN ONE LINKING OPTION!

In addition, linking *screenplay's* runtime unit is your choice! You can link *screenplay* by interrupt or directly to your application. And if your compiler doesn't allow a direct link, you can take advantage of a dynamic load option for linking *screenplay* to your application.

FULL CONTROL OVER KEY ASSIGNMENT!

You can assign practically any keyboard key to serve as a specific cursor function and define exactly which keys will return control to your application. *screenplay* gives you the power to entirely reconfigure the keyboard for your program.

POWERFUL, ONE-STEP PANEL PAINTING

screenplay's panel painting process is a "one-step" approach. There's no need to go through a separate process to establish fields on your I/O panel. What's more, you can use any ASCII character in your *screenplay* panels. You also have full control over character attributes such as foreground and background color, intensity and blinking.

EASY PANEL FILE MANAGEMENT!

Panels are stored in an ASCII file which is compressed to save memory and disk space. *screenplay's* Panel Management Facility allows you to easily copy panels across and within files, rename panels, delete panels, test and print panel details. You can even print an image of the panel for your documentation!

NO ROYALTIES / NOT COPY PROTECTED!

screenplay's panel control runtime unit, better known as The Panel Control Facility may be linked to your program without paying a dime in royalties. In addition, *screenplay* isn't copy protected to make it even easier-to-use. The Panel Control Facility allows you to control almost every panel characteristic by using parameters.

SUPPORTS MANY COBOL COMPILERS!

Supports IBM COBOL, Microsoft COBOL, Realia COBOL, Ryan-McFarland COBOL and Ryan-McFarland COBOL 8X; List Price \$175.00, OUR PRICE \$155.00



PROFESSIONAL TOOLS
FOR PROFESSIONAL
PROGRAMMERS

Quelo® 68000 Software Development Tools

Quelo Cross Assembler Packages are **Motorola compatible**. Each package includes a macro assembler, linker/locator, object librarian, utilities for producing **ROMable code**, extensive indexed typeset manuals and produces **S-records**, Intel hex, **extended TEK hex**, **UNIX COFF** and symbol cross references. **Portable source** written in "C" is available. It has been ported to a variety of mainframes and minis including **VAX, Sun, Apollo, Masscomp, and Charles River**. Native versions available for **Amiga** and **Atari ST**.

68020 Cross Assembler Package

Supports 68000, 68010, 68020, 68881 and 68851.
For CP/M-68K and MS/PC-DOS, \$ 750

68000/68010 Cross Assembler Package

For CP/M-80, -86, -68K and MS/PC-DOS, \$ 595

68000/68010 Simulator

For MS/PC-DOS by Big Bang Software, Inc.

SIM68K is a cost-effective software simulator for debugging 68000/68010 code on an IBM-PC or compatible, \$285.

Circle no. 302 on reader service card.

Trademarks: CP/M, Digital Research; MS, Microsoft Corporation; Quelo, Quelo, Inc.

HOURS
8:30 A.M. - 8:00 P.M. E.S.T.

800-421-8006

H E L P

is at hand

HELP/Control™ — an on-line help system for the IBM-PC. **HELP/Control** includes **HELP/Runtime**, **HELP/Popup** and our help screen compiler.

With **HELP/Runtime**, a few simple subroutine calls add context sensitive on-line help to your application. **HELP/Runtime** includes tested interfaces for C (Microsoft and Lattice), Pascal (Microsoft and Turbo), IBM BASIC (Interpreter and Compiler), Microsoft FORTRAN, COBOL (IBM and Realia) and assembler. Use our concise screen definition language to build your help files. You define the bold captions on your help screens and specify the links to other screens. If you have existing documentation files, it's easy to mark them up to get on-line quickly. You can put an entire user or reference manual on-line, completely accessible to the user at all times.

HELP/Popup provides memory resident access to your custom help screens.

The complete package (software, on-line manual, printed manual, and demo programs) costs \$125.00 and includes a royalty-free license to add **HELP/Runtime** to your applications and a license to make 25 copies of **HELP/Popup**.

Circle no. 303 on reader service card.



MDS, Inc. 207/772-5436

Double Your C Language Programming Productivity

Instant-C is an incremental compiler for C that makes every aspect of C programming as fast as possible. Load your existing C source code into **Instant-C**. Whenever you change your code with **Instant-C**'s built-in full-screen editor, only the changed portions of your programs are recompiled. The fully automatic compilation process includes linking of your multiple modules, and therefore **takes no link time**. **Instant-C** compiles more than twice as fast as normal compilers. More importantly, compile time is proportional to how much code you change, not to the size of your entire program. With **Instant-C**, you will be running your program within a few seconds after editing a change.

Instant-C's over 350 precise diagnostic messages help you understand immediately how to fix the problem. **Instant-C** shows any errors on the editor screen, with the cursor placed exactly at the trouble spot.

Instant-C runs your program at compiled speeds with full run-time checking of pointer references and array indexes. Run-time checking catches problems as they occur, when they are easiest to fix and understand. This checking reduces the number of times you need to modify and test your program.

Instant-C has the best testing and debugging capabilities available. You have full access to the C language and to your program. At any time you can examine or modify variables (including locals), evaluate expressions or macros, and call functions interactively. You can examine and change your code, then resume execution. Use any number of conditional breakpoints. **Instant-C**'s visual debugging shows your source code, highlighting the line being executed. **Instant-C** supports multiple screens, virtual screens, and non-standard graphics devices. **Instant-C**'s data monitor stops execution when specified variables or memory areas are changed.

Instant-C is compatible with Lattice 2.x, 3.x and Microsoft 3.0, 4.0. Get **Instant-C**'s superb debugging, run-time checking, and incremental compilation to double your C programming productivity.

Rational P.O. Box 480
Systems, Inc. Natick, MA 01760 LIST: \$495
Circle no. 304 on reader service card.

THE PROGRAMMER'S SHOP™
128-D Rockland Street, Hanover, MA 02339
Mass.: 800-442-8070 or 617-826-7531 12/86

STRUCTURED PROGRAMMING

(continued from page 126)

to conform with the standard Modula-2 syntax for absolute variables. More complicated types of absolute variables in Turbo Pascal may require more hacking. The translated program has its *FOR I := 1 TO J DO* loop edited into *FOR I := 0 TO J - 1 DO* to reflect the different string indexing schemes.

Creating Library Modules

This discussion of translating Turbo Pascal programs into Modula-2 is not complete without considering the strong modular nature of Modula-2. Turbo Pascal programs use included files, as well as chained code segments and overlays, to tackle large programs. Putting many of these possibly reusable routines in library modules is indeed a sound decision.

You can use the Translator to create library modules by following these steps:

1. Create a single complete file from all the related included files. By *complete*, I mean that it does not rely on any external or further declarations or code.
2. Enclose this code in an empty pro-

gram. Add the program heading and an empty main body section.

3. Compile the Turbo Pascal code with the compiler to make sure it is assembled correctly.
4. Process the correct Pascal code through the Translator. This gives you a Modula-2 version of your "do-nothing" program.
5. Apply any hand-coded editing required on the Modula code.
6. Decide what to export from the module, and create the DEFINITION and IMPLEMENTATION modules accordingly.

The next time you ask a Turbo Pascal programmer, "Parlez vous Modula-2?" expect the answer to be "TRUE."

Open Loops in Modula-2

You can use open loops in Modula-2 to prevent "telescoped" error handling code. Table 1, below, shows a skeleton Modula-2 program that tests three error conditions. Notice how telescoped the code is. Table 2, below, shows the same program, but this time an open loop is used to enclose the main body. Each tested error condition uses an *EXIT* statement, which allows the rest of the main body to be on the same level regardless of the number of error conditions tested.

The open loop ends with an *IF* statement that is guaranteed to exit. Nested open loops can be used in a similar way to control multilevel errors.

From Readers

I received a short utility program for the IBM PC written in Modula-2 from James Janney, of Salt Lake City, Utah. The program, shown in Listing Nine, page 88, displays the status of the Caps-Lock and Num-Lock keys and allows you to change them. This is useful with keyboards such as the KeyTronic, whose LEDs get out of sync when you run programs such as CrossTalk. Janney's utility rectifies this situation; the only other remedy is to reboot.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and disk format (MS-DOS, Macintosh, Kaypro).

DDJ

(**Listings begin on page 70.**)

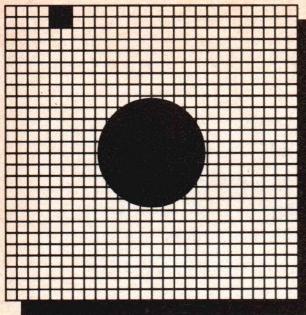
Vote for your favorite feature/article.
Circle Reader Service No. 8.

```
MODULE Test;
(* declarations *)
BEGIN
  (* Reset error flags *)
  ErrorCondition1 := FALSE;
  ErrorCondition2 := FALSE;
  ErrorCondition3 := FALSE;
  (* Statements *)
  IF NOT ErrorCondition1 THEN
    (* Statements *)
    IF NOT ErrorCondition2 THEN
      (* Statements *)
      IF NOT ErrorCondition3 THEN
        (* Statements *)
        END; (* IF NOT ErrorCondition3 *)
      END; (* IF NOT ErrorCondition2 *)
    END; (* IF NOT ErrorCondition1 *)
  (* Error handling section *)
  IF ErrorCondition1 THEN (* Statements *)
  ELSIF ErrorCondition2 THEN (* Statements *)
  ELSIF ErrorCondition3 THEN (* Statements *) END;
END Test.
```

Table 1: Telescoped code that performs error handling

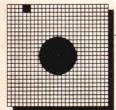
```
MODULE Test;
(* declarations *)
BEGIN
  (* Reset error flags *)
  ErrorCondition1 := FALSE;
  ErrorCondition2 := FALSE;
  ErrorCondition3 := FALSE;
  LOOP
    (* Statements *)
    IF ErrorCondition1 THEN EXIT END;
    (* Statements *)
    IF ErrorCondition2 THEN EXIT END;
    (* Statements *)
    IF ErrorCondition3 THEN EXIT END;
    (* Statements *)
    IF 1 > 0 THEN EXIT END;
  END; (* LOOP *)
  (* Error handling section *)
  IF ErrorCondition1 THEN (* Statements *)
  ELSIF ErrorCondition2 THEN (* Statements *)
  ELSIF ErrorCondition3 THEN (* Statements *) END;
END Test.
```

Table 2: Using an open loop for cleaner code



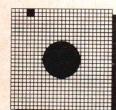
Better BASIC™

NOW INTRODUCING VIRTUAL MEMORY SUPPORT
BetterBASIC with the optional Virtual Memory Manager can
now address 400,000,000,000 bytes of memory!



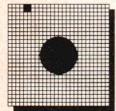
BetterBASIC Application Development System \$199.00

The BetterBASIC Application Development System provides very close compatibility with PC-BASICA and GW-BASIC, yet provides numerous new and sophisticated language features such as: program Block Structures, recursive Procedures and Functions with local variables, structures, Records and Pointers and last but not least support of large memory.



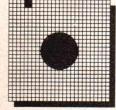
BetterBASIC Application Development System \$199.00

The BetterBASIC Application Development System provides very close compatibility with PC-BASICA and GW-BASIC, yet provides numerous new and sophisticated language features such as: program Block Structures, recursive Procedures and Functions with local variables, structures, Records and Pointers and last but not least support of large memory.



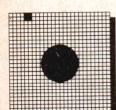
Virtual Memory Manager \$99.00

The Virtual Memory Manager expands Better-BASIC's data space into the gigabyte range and finally breaks the 640k byte barrier for array sizes. Not only can you directly address all expanded memory supported by LIM/EMS memory boards, you can also address any RAM Disk, Hard Disk or even a Floppy Disk as if they were ordinary RAM.



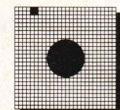
C-Link \$99.00

This software package allows BetterBASIC to access C-language library functions from within BetterBASIC. Currently supported are Lattice and Microsoft C.



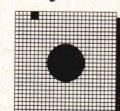
Screen Design System \$199.00

This package truly takes the drudgery out of creating display screens and data entry screens. An interactive Screen Editor lets you "paint" your display screens exactly as you want them to appear in your program. The completed screens take the form of disk resident images. A run time library module provides many new BetterBASIC procedures and functions for interacting with the display screens to simplify the use of pop-up menus and data entry screens.



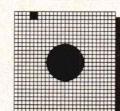
Btrieve™ Interface \$99.00

This is a high level BetterBASIC interface to the ever popular Btrieve™ file manager from SoftCraft. Instead of Assembly language calls this module provides high level BetterBASIC program access to all Btrieve™ functions. Use it to design your own database application in BetterBASIC.



8087/80287 Math Module \$99.00

This module allows you to use the 8087 or 80287 co-processor to significantly accelerate programs which are floating point calculations intensive.



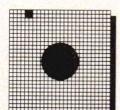
Decimal Math Module \$99.00

If you are a business programmer, you are probably frustrated by the many roundoff problems caused by ordinary IEEE format floating point numerical operations. The BetterBASIC Decimal Math Module which offers variable precision from 6 to 24 digits, drastically reduces roundoff problems in business applications.



BetterTools™ \$99.00

This is a collection of more than 150 useful extensions to BetterBASIC such as time and date computations, encryption and decryption, low level file directory access, hyperbolic function and much more. No BetterBASIC programmer should be without BetterTools™.



Virtual Memory Manager Network Version \$250.00

This version of the Virtual Memory Manager allows Virtual Memory to be distributed throughout a Local Area Network. It also provides File, Records and Field Locking to control access to shared data.

Call our Toll Free Order Line
1-800-255-5800

Better
BASIC.

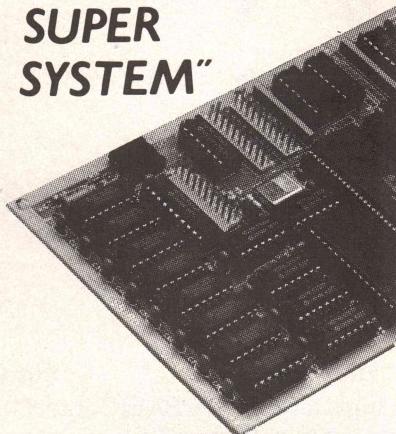
Summit Software Technology, Inc.™

106 Access Road, Norwood, MA 02062

Circle no. 363 on reader service card.

Byte Magazine called it.

"CIARCA'S SUPER SYSTEM"



The SB180 Computer/Controller

Featured on the cover of Byte, Sept. 1985, the SB180 lets CP/M users upgrade to a fast, 4" x 7½" single board system.

- **6MHz 64180 CPU**
(Z80 instruction superset), 256K RAM, 8K Monitor ROM with device test, disk format, read/write.
- **Mini/Micro Floppy Controller**
(1-4 drives, Single/Double Density, 1-2 sided, 40/77/80 track 3½", 5¼" and 8" drives).
- **Measures 4" x 7½", with mounting holes**
- **One Centronics Printer Port**
- **Two RS232C Serial Ports**
(75-19,200 baud with console port auto-baud rate select).
- **Power Supply Requirements**
+5V +/- 5% @500 mA
+12V +/- 20% @40mA
- **ZCPR3 (CP/M 2.2/3 compatible)**
- **Multiple disk formats supported**
- **Menu-based system customization**

SB180-1

SB180 computer board w/256K bytes RAM and ROM monitor

\$299.00

SB180-1-20

same as above w/ZCPR3, ZRDOS and BIOS source.....

\$399.00

-Quantity discounts available-

NEW

COMM180-M-S

optional peripheral board adds 1200 bps modem and SCSI hard disk interface.

TO ORDER
CALL TOLL FREE
1-800-635-3355

TELEX
643331

For technical assistance or
to request a data sheet, call:

1-203-871-6170



Micromint, Inc.
4 Park Street
Vernon, CT 06066

FORUM

LETTERS

(continued from page 12)

Eratosthenes Sieve

Dear DDJ,

Regarding the Eratosthenes Sieve that accompanied the article "High-Speed Thrills" in the September 1986 issue, the program reports 1,899 primes in the range 2 to 8,191. This is incorrect. Actually, there are only 1,028 primes in that range.

I realize that the Sieve is used as a benchmark to test a system's ability to handle arrays and loops and that the actual number returned is immaterial. Still, it bothers me that people may be going around thinking there are 1,899 primes less than 8,192. Code Example 1, below, returns the correct result. It runs just a little slower than the listing you published (after

all, it's filtering out more nonprimes). I've also included a loop at the end of my program that prints out the primes, just to prove that the program does find primes. This should be removed if the program is to be used as a benchmark.

Robert V. Duncan
P.O. Box 215
Pottersville, NY 12860

DDJ

```
#define TRUE 1
#define FALSE 0
#define SIZE 8191
char flags[SIZE+1];

main()
{
    int i, j, count, loops;
    printf("Running . . . (TC)n");
    for (loops = 0; loops < 10; loops++)
    {
        count = 0;
        for (i = 2; i <= SIZE; i++)
            flags[i] = TRUE;

        for (i = 2; i <= SIZE; i++)
        {
            if(flags[i]) /* found a prime, */
            {
                for (j = 2*i; j <= SIZE; j = i)
                    flags[j] = FALSE; /* "filter" out its multiples
*/                count++; /* increment prime
counter */
                /* endof for j = . . . */
            } /* endof for i = 2 . . . */
        } /* endof for loops == 0 . . . */
        printf("%d primes, %d loops (TC)n", count, loops);
        /* let's take a look at our primes
        zero and one aren't prime, start at 2
*/
        for (i = 2; i <= SIZE; i++)
            if (flags[i])
                printf("%d", i);
        print("(TC)n");
    }
}
```

Code Example 1: Eratosthenes Sieve prime number program in C

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.

- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.



SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- the C compiler for MVS software developers
- the C compiler for CMS software developers
- the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA.
27511-8000. Telephone (919) 467-8000, x 7000

DDJ 2/87

VIEWPOINT

(continued from page 14)

Total cost \$4.40

Item onions

Quantity 2.5 lbs

Unit cost \$0.72/lb

Total cost \$1.80

Would you not be somewhat puzzled about the rationale for that arrangement? I imagine that you would instinctively attempt to rearrange the list as shown in Table 1, right. Surely this is clearer, more obvious, easier to read, and less likely to hide an inadvertent error. In fact, isn't the arrangement in Table 2, right, even better?

Assuming you agree, why then in Anderson's Modula-2 program are there 118 repetitions in the following form?

WITH Table68K [i] DO

```
Mnemonic := "ABCD";
Op := {15, 14, 8};
AddrModeA := ModeA {Rx911,
                     RegMem3, Ry02};
AddrModeB := ModeB {};
END
```

At the very least we should expect that 118 repetitions of the same structure could be reduced to something such as:

```
Table68K[i] := "ABCD", {15, 14, 8},
                {Rx911, RegMem3, Ry02}, {};
```

Surely our "higher order" language should allow us the benefits of parallel construction and avoid the need for excessive repetition of unnecessary words. With only the smallest dash of literary license, we really ought to expect something such as the format shown in Code Example 2, right. To me this tabular form seems cleaner, much easier to comprehend, and much more likely to show up problems. The parallel construction encourages comparison and highlights errors. And even without these advantages, the result is at least 7,000 characters shorter and uses about 590 fewer lines, which, if nothing else, saves ten pages of printout.

The point is this: Although Modula-2 has been designed to "force" on programmers what is considered "best" form, it does not allow them to

be as clear, clean, succinct, and precise as they would almost instinctively be without it. As a proof without comment, Code Example 3, below, offers an example of what a programmer would have done had he or she been forced to write the preceding code in assembly language.

Blunted tools produce crude carvings, and furry language fosters fuzzy thinking. So if my comments

about style and form are cogent, we ought to be able to find some deficiencies in Anderson's code.

There are a couple of rather odd things about this code. As I understand it, it is used only to create a table of data on disk. It does this by building up the entire table in memory, one record at a time, and then writing the entire table out to disk, one record at a time. I argue that it

| Item | Quantity | Unit Cost | Total |
|----------|----------|-----------|--------|
| potatoes | 3.2 lbs | \$0.65/lb | \$2.08 |
| oranges | 5.0 lbs | \$0.88/lb | \$4.40 |
| onions | 2.5 lbs | \$0.72/lb | \$1.80 |

Table 1: A possible rearrangement of a grocery list

| Item | Quantity (lbs) | Unit Cost (\$/lb) | Total (\$) |
|----------|-------------------|----------------------|---------------|
| potatoes | 3.2 | 0.65 | 2.08 |
| oranges | 5.0 | 0.88 | 4.40 |
| onions | 2.5 | 0.72 | 1.80 |

Table 2: An improved rearrangement of a grocery list

| Table68K := | | | |
|-------------|--------------------------|---------------------------|-----------|
| Name | Op | AddrModeA | AddrModeB |
| "ABCD" | {15, 14, 8} | {Rx911, RegMem3, Ry02} {} | |
| "ADD" | {15, 14, 12} | {OpM68D} | {OpEA05y} |
| ... | ... | ... | ... |
| "UNLK" | {14, 11, 10, 9, 6, 4, 3} | {Ry02} | {} |
| : | | | |

Code Example 2: A tabular format of Anderson's code

```
Table68K:
  db 'ABCD'
  dw 1100000100000000b, 000000000000111b, 0000000000000000b
  db 'ADD'
  dw 110100000000000b, 000000100000000b, 0010000000000000b
  ...
  db 'UNLK'
  dw 0100111001011000b, 00000000000010b, 0000000000000000b
```

Code Example 3: An assembly-language version of Anderson's code

would be better to write out each record as it is created, thereby requiring storage for only one record rather than for 118 of them. This would give a shorter program, use less memory, run much faster (because no array indexing would be required), and take no longer to compile.

Even odder, Anderson's current code builds a data structure at execution time in a situation where all the data that goes into the structure is contained in the code at compile time. This is a little oblique. It would be more reasonable to declare the structure as an initialized constant. This would give an even smaller program that would run still faster and take even less time to compile.

I do not call these peculiarities errors, although for programmers they are further from good programming than we would allow writers to stray in their English. My thesis is that they are more the fault of the language than of the programmer. Indeed, the worst is actually forced: in Modula-2 there is no way to initialize a constant array.

I believe that, in our attempts to design programming languages that so constrain programmers that they are "unable to write bad code," we are removing them so far from what is actually going on that they are often not even conscious of the absurdities they are creating.

It is odd, and we ought to reflect on it, that in the case of Modula-2, which has been most heralded as a step toward "improving" our programming, we regularly find the most blatant variances between what is considered "approved practice" and the intuitive dictates of our linguistic common sense.

Is there not something wrong with the direction in which we are being led?

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 1.

Tom Rettig's Library™

Prewritten Solutions to Programming Problems

Clipper Edition

Advanced Extended Library

- Convenient .LIB library file to link
- Separate object files so only the code actually used is added to your program
- Optimized for speed and code size
- Requires Clipper, Winter '85 or later

Each edition comes complete with...

- Entire source code, nothing withheld. Cleanly written, liberally commented, easy to modify, helpful for learning.
- Support by phone and electronic mail
- No royalties or copy protection

dBASE III PLUS Edition

Advanced Programmer's Library

- Interface to C/assembler is fully compatible with Clipper's Extend system
- Functions are CALL procedures
- Requires dBASE III PLUS, any version, and 128K of additional memory

Over 175 functions; 55% written in C, 35% in Assembler, 10% in dBASE

\$99.95 per edition at dBASE/Clipper dealers or direct



**Tom Rettig
Associates**
Excellence in dBASE since 1982

9300 Wilshire Boulevard, Suite 470
Beverly Hills, CA 90212-3237, U.S.A.
(213) 272-3784 ■ Telex: 4996426 RETTIG
Source: BCR480 ■ CompuServe: 75066,352

dBASE and dBASE III PLUS are Ashton-Tate trademarks, Clipper is a Nantucket trademark

Call or write for free product information

Circle no. 175 on reader service card.

Feel Constrained By PC BATCH Languages?

You need OPAL, a rich, full-functioned executive shell language for the experienced programmer.

Previously, you've had only limited capabilities with the batch exec languages available on the IBM PC. Now, OPAL frees you from the limitations and drudgery of BATCH, and other primitive executive languages.

OPAL, as an interpretive language, makes it easy to program the control you need in your applications. And, OPAL's compiler gives you even greater machine efficiency, when required.

Functionally, OPAL is a flexible shell that offers a high degree of programming, featuring flow-of-control, Menu and Form screen definition, CALLS, DO statements, DATE and string functions, to name a few. OPAL is so complete, some users have used it for prototyping entire applications!

Circle no. 361 on reader service card.

Moreover, OPAL provides benefits through its synergy with DOS. OPAL understands how to work with and enhance your DOS environment.

If you feel encumbered and constrained by your current exec language, you NEED OPAL...

\$169
...only

Texas residents add 6.125% sales tax
Shipping \$4.00
MasterCard, VISA Accepted

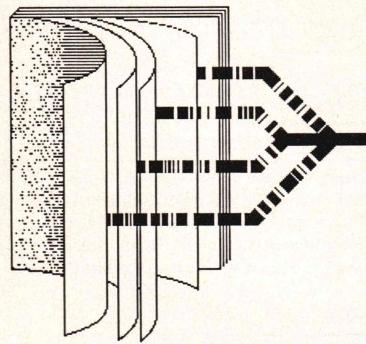
To order or receive more information:

(214) 490-0835

The Software Factory
INCORPORATED

15301 Dallas Parkway
Suite 750 LB 44
Dallas, TX 75248

IBM is a registered trademark of International Business Machines Corp.
OPAL is a trademark of The Software Factory, Inc.



PROLOG and the Future of AI

The following is a continuation of the excerpt of a real-time conference that we published in January. The conference was held by Borland International on CompuServe on July 26, 1986. A complete transcript of this three-hour on-line conference on AI and PROLOG can be found in DL6 of the Borland SIG on CompuServe (<GO BOR100>KEYWORDS:CONFERENCE).

Larry Kraft, sysop of Borland SIG: Our panel of featured "speakers" today includes Borland's president, Philippe Kahn; assistant professor Mark Chignell of USC; and Mike Swaine, editor-in-chief of *Doctor Dobb's Journal of Software Tools*. . . . Here's a question that sounds straightforward: What are theorem-proving algorithms?

Philippe: Nothing, until now! The idea is that you could set up a system that would be a sort of auditing trail through algorithms—in the same way as people have tried to do automatic proofing of mathematical theorems. It's great in the ivory tower but useless in proving or finding new theorems. The four-color problem was not solved that way but rather through brute force. Some people have even questioned the validity of the proof because the algorithms used could not be proved thoroughly.

Mark: We should make it clear that we are not saying it's not useful to prove theorems. Logic programming requires theorem proving, of a sort, every time you make an inference. It's just the attachment of the auto-

matic verification that is a problem.

Philippe: That's right.

Larry: One more question before we proceed with questions from the floor: How is CD ROM technology going to impact AI? And, what's the future of the PC and how will that impact AI?

Philippe: Well, the biggest limitation of PCs now is the data storage size. The bulk of a knowledge base can be read-only because the machine seems now to have enough read and write storage capacity. Distributing large knowledge bases on CD ROMs in my opinion will greatly enhance the use of expert systems on PCs. Accessing the information is another unrelated question. The drivers have to be optimized to the type of data structure used when mastering the CD ROM because keyword searches are inefficient for most AI applications. CD ROMs are going to add a lot to the power of the PC.

Larry: Any comments from Mike or Mark before we conclude the panel phase of this conference and proceed with questions from the floor? OK, I'll take questions from the floor now. Go ahead Dan.

Dan: Philippe, why would anyone designing an expert system want to use PROLOG rather than one of the existing tools such as 1st Class, Insight2+, or some of the more sophisticated KEE/M.1 type of tools?

Philippe: It's the idea of control. Borland will release—it's no secret—some prebuilt shells that work on top of our implementation of PROLOG. But what really made the success of products such as dBASE from Ashton-Tate has been their programmability. And that is what you get when you start with the underlying implementation language. You have the control and the full source!

Dan: I understand and agree on the control issue, but it seems to me that many people who will want to design expert systems are experts

themselves, rather than designers. I'm using PROLOG for my own work, but I know a lot of people who aren't and I frequently have the argument, which is why I asked the question.

Philippe: Yes, that's true. That's why we are working on additional tools. Take the example of dBASE. For an accounting system, for example, you need both the expert—the CPA—and the dBASE programmer if you really want to build something useful. As a matter of fact, it is the same thing with most spreadsheets that involve programming in some sense—if you do not wish to end up with a monster!

Mark: I'm all for experts designing and implementing their own systems. It may not be as tough as people think. I happen to think that high-school algebra is more difficult to learn than the basic technology of expert systems. I ran a course this summer at USC, and we had computer neophytes building rudimentary expert systems in PROLOG. A continuum of utilities can be added on top of PROLOG to support expert system development. Why limit yourself to the idiosyncrasies of a single shell?

Larry: Chris is next. Go ahead, please Chris.

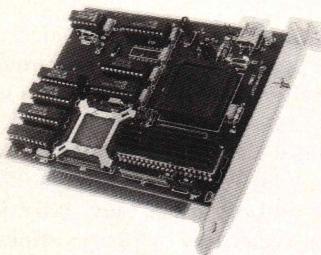
Chris: With reference to the Fifth Generation Computer Project, Terry Winograd and Fernando Flores, in their book *Understanding Computers and Cognition* (Ables, 1986), said: "The grandiose goals, then, will not be met, but there will be useful spinoffs. In the long run, the ambitions for truly intelligent computer systems, as reflected in this project and others like it around the world, will not be a major factor in technological development. They are too rooted in the rationalistic tradition and too dependent on its assumptions about intelligence, language, and formalization." Will the panel please comment?

Philippe: Do they mean that the machine is always going to remain dumb but there will be some real useful technology that will come out of it? Just like the space shuttle?

MICROWAY MEANS 8087 PERFORMANCE

FastCACHE-286™

Runs the 80286 at 8.5 or 11 MHz and the 80287 at 5, 6 or 11 MHz. Includes 8 kbytes of 55ns CACHE Works with more PCs than any other accelerator, including Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache and Diagnostics..... From \$449



DATA ACQUISITION and REAL TIME TOOLS

DAL™ – "Data Acquisition Language." **Unkelscope™** – A real time data acquisition, control and process software pkg.

87FFT and 87FFT-2

TransView Menu driven FFT Spectrum/transfer analyzer \$250

RTOS - REAL TIME OPERATING SYSTEM

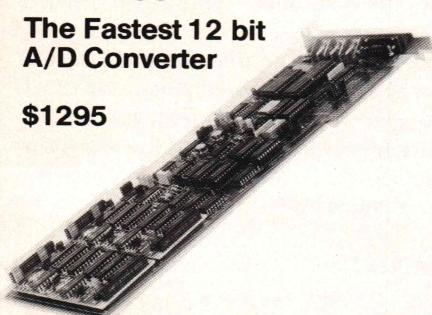
A multi-user, multi-tasking real time operating system. Includes a configured version of Intel's iRMX-86, LINK-86, LOC-86, LIB-86, OH-86 and the MicroWay 87DEBUG. Runs on the IBM-PC, XT, PC-AT and COMPAQ \$600

INTEL COMPILERS Available for RTOS
FORTRAN-86, PASCAL-86, PL/M-86.

A2D-160™

The Fastest 12 bit A/D Converter

\$1295



160,000 Samples per second
Pseudo Random Noise Generator/DAC
Optional signal conditioners
AFM-50 Programmable Low Pass Filter Module..... \$225

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

**Micro
Way**

P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

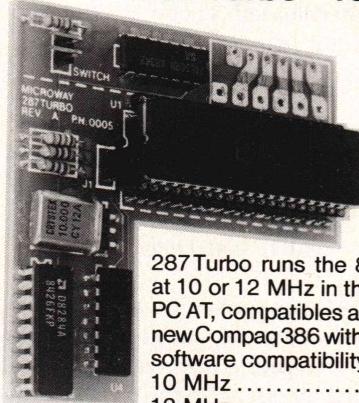
LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles...\$549

MegaPage with ØK..... \$149

MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. Sold populated with 1 megabyte CMOS ... \$699 or with 3 megabytes CMOS cool running low power drain RAM ... \$1295. Optional serial/parallel daughterboard..... \$95

287 Turbo™-10/12



287 Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz \$450
12 MHz \$550

PC Magazine "Editor's Choice"

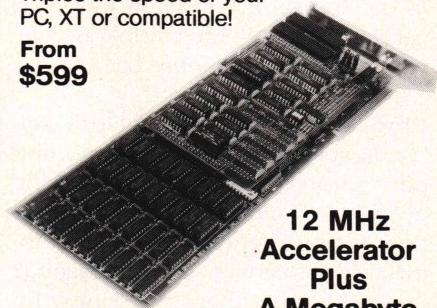
8087 SOFTWARE

| | |
|----------------------------------|-------|
| IBM BASIC COMPILER | \$465 |
| MICROSOFT QUICK BASIC | \$79 |
| 87BASIC COMPILER PATCH | \$150 |
| IBM MACRO ASSEMBLER..... | \$155 |
| MS MACRO ASSEMBLER..... | \$99 |
| 87MACRO/DEBUG | \$200 |
| MICROSOFT FORTRAN..... | \$209 |
| RM FORTRAN..... | \$399 |
| LAHEY FORTRAN F77L..... | \$477 |
| MS or LATTICE C | CALL |
| STSC APL★PLUS/PC..... | \$450 |
| STSC STATGRAPHICS | \$675 |
| SPSS/PC+..... | \$675 |
| 87SFL Scientific Functions | \$250 |
| PHOENIX PRODUCTS..... | CALL |
| FASTBREAK for 1-2-3 V.1A | \$79 |
| HOTLINK for 1-2-3 V.1A | \$99 |

NUMBER SMASHER/ECM™

Triples the speed of your PC, XT or compatible!

From
\$599



12 MHz Accelerator Plus A Megabyte for DOS

PC Magazine "Editor's Choice"

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz \$114
For the IBM PC, XT and compatibles

8087-2 8 MHz..... \$149
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5 MHz..... \$179
For the IBM PC AT and 286 compatibles

80287-6 6 MHz..... \$229
For 8 MHz AT compatibles

80287-8 8 MHz..... \$259
For the 8 MHz 80286 accelerator cards

80287-10 10 MHz..... \$395
For the Compaq 386

**Call for prices on V20, V30,
64K, 128K and 256K RAM**

287 TURBO-PLUS™ Speeds up your AT

Adjustable 80286 Clock 6-12 MHz
10 MHz 80287 Clock
Plus Full Hardware Reset..... \$149
Optional 80286-10



287TURBO-PLUS
With 80287 10 MHz..... \$549
With 80287 12 MHz..... \$629

**The World Leader
in 8087 Support!**

MicroWay Europe
32 High Street
Kingston-Upon-Thames
Surrey England KT1 1HL
Telephone: 01-541-5466

Mark: It's getting harder to separate our technology from our intellectual theories. I happen to think that the rationalist approach may have a chance in the future. If you want to see an earlier view, which I think is relevant, see the philosopher Leibniz.

Chris: Well, Winograd and Flores say: "Gadamer, Heidegger, Habermas, and others argue that the goal of reducing even 'literal' meanings to truth conditions is ultimately impossible and inevitably misleading." And on page 132 they add: "First there is a danger inherent in the label 'expert system.' When we talk of a human expert we connote someone whose depth of understanding serves not only to solve specific well-formulated problems, but also to put them into a larger context. We distinguish between experts and 'idiot savants.' Calling a program an expert is misleading in exactly the same way as calling it 'intelligent' or saying it 'understands.'"

Philippe: OK, let's stop calling these programs "expert systems" and call them "advisory systems."

JRCOOPER: I'd like to direct a question to Philippe. It would appear that LISP is being considered the language of the past and that PROLOG is the language of choice for the future. Now, hybrid languages aside, there must be a pretty good reason why you decided to choose PROLOG over other AI languages. I recall an interview you did for *Computer Language* magazine last year in which your "vocal disdain for C" was mentioned on the same page as it was said you had no plans for an AI language. The real question here is why PROLOG and not LISP or Smalltalk?

Philippe: Well, we've already said use the right tool for the right purpose. To answer the rest of your question, I usually contradict myself a lot. My wife hates it! Also, I don't disdain C, but I don't think it is a good hobbyist language. I believe that Pascal and Modula-2 are much better and more readable. In order to program well in C, you really need to know what you are doing. Pascal is more typed and structured for that purpose, and its

nested structures, in my opinion, offer more readability than C's flat procedure and function structure.

JRCOOPER: OK, but was there a conscious decision *not* to introduce a Borland LISP or Smalltalk?

Philippe: Smalltalk is a truly great language, and its only drawback is that it's big on a PC. No, there is another factor here, which is competitiveness with other language houses. We felt that PROLOG would open the "brave new world of AI" to our users and ourselves. That doesn't mean that we are not working on other languages. Furthermore, we wanted to add a new flavor to a market that was becoming boring. I think that, had we not released Turbo Prolog, we would not have all this stir about AI right now!

MFE: In your experience, is it difficult for veterans of the old high-level languages to pick up PROLOG? I seem to be having problems. And, what's the best way to learn PROLOG?

Mark: My students who are familiar with procedural languages have trouble "thinking backward" in PROLOG, as they put it. On the other hand, the transfer between LISP and PROLOG appears to be good because of common data structures and the use of recursion. I would suggest the following readings. First, *A PROLOG Primer* by Jean Rogers. After that, look at what has become the classic—*Programming in PROLOG* by Clocksin and Mellish. I would also recommend two articles in *Communications of the ACM*, December 1985, and the August 1985 issue of *Byte*.

Mike: I'd also like to modestly recommend a *DDJ*, March 1985, article by David Cortesi for PROLOG beginners. And Clocksin and Mellish, of course. I conducted a little experiment in which I tried teaching PROLOG to a programmer and to a nonprogrammer. The nonprogrammer had much less trouble catching on.

MFE: P.S.—I'm just glad I can be floundering around with PROLOG without having to take out a second

mortgage! Keep up the good work Borland.

Larry: Go ahead Hannu.

Hannu: I think we need more hardware advancements before we can have true AI. How about it panel? We still only have 1s and 0s?

Philippe: You will probably still have 1s and 0s a few years from now. But I always say software is ten years behind hardware. I really think that it is now a software technology problem. A lot of exciting things can be done without much new hardware. It is a weakness of programmers in many cases to think that they are limited by the hardware. In France we say "a bad worker seems always to have the wrong tools!"

Mark: Did you see the TI satellite symposium recently? I take great objection to the idea that you have to get the best of everything to be able to do serious AI. Have you seen the segment on "Saturday Night Live" on the limits of the imagination? My feeling is that the main problem is the logic and reasoning itself rather than the grunt of the machine. So I guess I'm backing up Philippe's comment. I think that good AI applications can be built on the AT, but maybe you have to think a bit more than letting a LISP machine set all the defaults for you.

Hannu: I agree that the hardware is way ahead, but we do need the hardware to keep up with the goals we are trying to reach! And these are truly self-pacing—learning and modifying codes and so on.

Mike: New hardware is always fine, but it's just hardware. A great chef can always use a new pot, though.

Philippe: Yes, but the oldest pots tend to give the best flavor. Have you tried Teflon lately?

Larry: I'm going to let MikeM in here.

MikeM: Thanks. I was wondering what all you *users* of PROLOG have been developing using the new package?

80386

SOFTWARE DEVELOPMENT TOOLS

The Phar Lap 80386 Software Development Series:

386|ASM/LINK by Phar Lap (MS-DOS®) \$495

Full-featured macro assembler and linker. Includes a debugger and 32-bit protected mode runtime environment.

80386 High-C™ by MetaWare (MS-DOS®) \$895

80386 Professional Pascal™ by MetaWare (MS-DOS®) \$895

UNIX™ and VAX/VMS® cross tools (call)

The wait for professional software development tools for the 80386 is over! Whether you are upgrading an existing IBM PC application to the '386, moving a mainframe application down to a PC, or writing the next PC best-seller, the Phar Lap 80386 Software Development Series is for you. It is an integrated line of products which provides everything you'll need to create and run 80386 protected mode applications under MS-DOS.

(617) 661-1510

Phar Lap Software, Inc. "The 80386 Software Experts"

60 Aberdeen Ave.

Cambridge, MA 02138

Circle no. 343 on reader service card.

'C' THE BEST METACOMCO'S LATTICE

C



"C" COMPILER
FOR ATARI ST
and AMIGA

Full implementation of
Kernighan and Ritchie —
Based on Lattice's very successful
range of 8086/88 C Compilers.

—also available—

Macro Assembler - Professional development system ST - \$ 79.95
Amiga - \$ 99.95

BCPL - NEW! Full standard BCPL compiler - ST \$149.95

Lattice 'C' - The well known Lattice 'C' compiler - ST \$149.95

Cambridge Lisp - The interpreter/compiler - ST & Amiga \$199.95

MCC Pascal - Fast ISO/ANSI standard compiler - ST & Amiga \$ 99.95

Metacomco MAKE - NEW! UNIX-like MAKE utility for the ST \$ 69.95

MENU+ - Best selling ST MENU generator \$ 29.95

Metacomco SHELL - NEW! Amiga's intelligent programming shell \$ 69.95

Metacomco TOOLKIT - Smartest tools available for the Amiga \$ 49.95

Cambridge LISP - CPM-68K - \$295. Call for Sinclair QL products. Languages come with full documentation, libraries & screen editor. ST languages include MENU+ and provide full interfaces to GEM VDI and AES functions. Metacomco provides experienced technical support and keeps its customers informed of new products and upgrade releases.

METACOMCO

5353 #E Scotts Valley Dr.
Scotts Valley, CA 95066

Registered trademarks: Lattice - Lattice, Inc; Atari ST -
Atari; UNIX - Bell Labs; Amiga - Commodore Amiga.

Contact your local dealer or call:
Tel: (US) 800-AKA-META (CA) (408) 438-7201
BIX: mhll Compuserve: 73247,522
Add 6½% tax if CA resident



Circle no. 358 on reader service card.

PC/VI™

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI** — a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

PC/VI is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI**:

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Auto-indent and Showmatch
- MUCH, MUCH MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI". "The documentation is so good I have already learned things about VI that I never knew before." — IEEE Software, September 1986.

PC/VI is available for IBM-PC's and generic MS-DOS® systems for only \$149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

PC/TOOLS™

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. These tools were designed and have been continually enhanced over the last fifteen years! Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

- | | | | |
|--------|---------|-------|-----------|
| • BFS | • DIFFH | • OD | • STRINGS |
| • CAL | • DIFF3 | • PR | • TAIL |
| • CUT | • GREP | • SED | • WC |
| • DIFF | • HEAD | • SEE | |

All of these for only \$49.00; naturally, extensive documentation is included!

PC/SPELL™

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only \$49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** now and get **PC/SPELL** for only \$1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 8", 5½" and 3½" disk formats. For more information call today!

*UNIX is a trademark of AT&T. MS-DOS is a trademark of Microsoft.

CUSTOM SOFTWARE SYSTEMS

P.O. BOX 678 • NATICK, MA 01760

617 • 653 • 2555



Circle no. 268 on reader service card.

Mark: So far I've been using it in an educational environment. Most of my students work in the aerospace industry in Southern California. They are building a variety of small expert systems in such areas as monitoring data from a satellite, developing systems to aid instrument landings in aircraft, and advising safety engineers on how to collect data. We are trying to build a frame representation language, but it has proven tough with the typing that Turbo Prolog uses.

Larry: We have a comment from Dan Kernan, one of our technical support people for Turbo Prolog.

Dan K.: Strong typing should not cause a constraint on a frame-based system. A frame is generally a structure with attributes and can be constructed easily within our typed system.

Larry: DanS, You're up next. Go ahead, please.

DanS: My question is directed first to Mark, then to the others if they like. Can you see any micro implementation of PROLOG being powerful enough to sit in the background of existing programs' databases, spreadsheets, or other applications and to invoke themselves by sensing when their users are in trouble? The problem I see with today's "help key" approach is that users almost never know they *need* the help. Any comments on what role PROLOG and micros might play here?

Mark: I think Turbo Prolog is moving toward the kind of power you are talking about. The availability of a fast compiler is a big plus. I suspect that the best configuration may be to have PROLOG sitting on top of a retrieval engine. We're looking at this at USC, and I know that Philippe has more to say on this topic.

Philippe: Well what you are really talking about is a "lightning" type of system where something watches in the background and tries to help you—like asking "Do you really mean this, or do you mean these things?" It

is more programming style than an implementation language. As I said before, before it's done it's often called AI, once it works it's called a program.

JRCOOPER: This is for all panelists. There seems to be an interest in hybrid languages that to some small degree address the old "best tool for the given application" issue. Now we are getting LISP systems that allow for object-oriented programming, Smalltalk systems that come with PROLOG-like inference engines, and so on. Does anyone feel that there could someday be a language that provides a proverbial AI Swiss-army knife?

Philippe: There really should not be any fanaticism in the choice of one implementation language rather than another. As a matter of fact, that choice is becoming a whole new field of study: "What is the optimal implementation language for a given application?" In the same way, maybe French is better for poetry, English is better for oral communications, and German for technicalities.

Mark: It's going to get to the point at which it's not clear what the boundaries between languages are. I don't see why we shouldn't use the family of languages concept with a certain amount of specialization in the functions of each language. This seems particularly suited to likely developments in concurrent processing and parallel machines.

Philippe: But the danger of not having specific tools is that of Ada. I don't believe you can efficiently be everything to everybody!

Mike: The only reason there are Swiss-army knives is that pockets are of limited size. That doesn't apply to computers, ultimately.

JRCOOPER: No, but it does to my wallet.... One final question—I first started reading about AI around the turn of the decade. I was just getting into computers back then, and at that time the research in AI seemed to be toward literally making the machine think like we do. There were all kinds

of theories about how our brains work and so on. It now seems like AI has become more pragmatic and seems to be focused primarily on expert systems. Has the AI community dropped its lofty goals of a decade ago?

Philippe: Well, you are right. At one time people thought that natural language interfaces would be the thing. It turned out, however, that the resulting commercial products—such as Clout, for example—never really captivated a following. Why? Because typically the people using a database system and querying it with the help of a natural language query system did not know how to type really fast and a short-cut approach was more efficient. So as long as continuous speech recognition is not commercially available at a reasonable price, this is yet a dream. Expert systems are a reality and are useful, however. That is why they are now the predominant part of commercially oriented AI applications.

Mark: Expert systems are a commercial application of what used to be AI. But expert systems work is limited and brittle. What I find intriguing is the recent work in machine learning, where it looks like we may be getting a little closer to understanding induction, generalization, and learning by analogy. This is a very open-ended area and to my mind the truest of AI domains.

Philippe: Oh, I agree 100 percent. This is the most exciting part!

Mark: AI has a business and a research side to it. Sometimes the two tend to get confused! Expert system applications have been getting all the press, but people haven't forgotten the basic goals. There is interesting work on machine learning, and there are plenty of cognitive scientists, linguists, and so on who are using the tools and concepts of AI to explore the fundamental issues in the acquisition and utilization of intelligence.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 9.

PLOTDEV ADDS GRAPHICS TO PC-DOS \$39

MicroPlot, innovator of creative PC software programs is excited to announce PlotDev, the alternative to virtual device drivers and graphics tool kits for PC-DOS graphics programs.

- Installable PC-DOS device driver adds graphics command capabilities to any program language
- Allows full screen PC-DOS command editing by unclicking the cursor
- Supports most popular graphics boards and provides a built-in graphics screen dump for printers
- Non-interfering with memory resident or application programs

- Provides user with all the intelligent alpha terminal commands of a DEC VT-100
- Provides user with the graphics commands of the Tektronix 4010/4014 and 4027 graphics terminals
- And much more ...

Site license and educational discounts available.

For a detailed PlotDev brochure or for ordering information call toll free 1-800-654-1217. Visa and MasterCard welcome.



MicroPlot

TM

659-H Park Meadow Road Westerville, Ohio 43081 (614) 882-4786

Circle no. 84 on reader service card.

The Latest Computer Science . . . from Springer-Verlag

The Art of C Programming R. Jones and I. Stewart

This new tutorial introduction to programming in C is intended for home-users and students who are new to the language. The book assumes a familiarity with a high-level language such as BASIC, and uses this as a starting point for discussing the distinctive features of C. Included are "case studies" of various aspects of the language. Clearly and humorously written, *The Art of C Programming* provides complete coverage of this versatile language.

1987/approx 224 pp./42 illus./softcover \$18.50
ISBN: 0-387-96392-8

Software Engineering in C P. Margolis and P. Darnell

This new introduction and reference manual for C contains many real-life examples and applications as well as the new ASCII C standard. *Contents:* Introduction, C Program Structure, Data Types, Statements, Operators, Arrays and Pointers, Structures and Unions, Storage Classes, Functions, Preprocessor, Input and Output, Runtime Library, Appendices.

1987/approx 350 pp./50 illus./softcover
(Springer Books on Professional Computing)
Available: June 1987

Taming the Tiger

Software Engineering and Software Economics
L.S. Levy

A new approach to software development with an economic justification based on the analytical techniques of managerial economics. The new approach modifies the life cycle of software development. Will be of special interest to researchers in economic theory as well as managers of software development and computer scientists.

1987/approx 256 pp./9 illus./softcover \$28.00
(Springer Books on Professional Computing)
ISBN: 0-387-96468-1

Ada* is a registered trademark of the U.S. Government,
Ada Joint Program Office

Circle no. 236 on reader service card.

program with power!

PC/POWER™

APPLICATION DEVELOPMENT & MANAGEMENT
INTRODUCTORY OFFER: \$95 Including s & h

Runs on all 100% PC compatibles!

- Supports applications in a variety of languages through program calls
 - NOT a code generator
 - C, PASCAL, BASIC, ASSEMBLER
- Screen painter/editor - create language and program independent data entry/display screens
 - Alphanumeric, Integer, Long, Floating point fields
- Build your own POP-UP menu/selection windows under your program control
- Supports CGA, EGA, and monochrome display adapters
- Indexing function for cataloging programs and applications provides easy management of ALL of your PC programs and packages.
 - Integrate existing programs into an application
- Supplies EXEC function to pass control between programs
 - Even pass data between programs in different languages
- Development utility applications included as samples
- Entire package written in assembler; lightning FAST!

COMPLETE DEVELOPMENT & RUNTIME SYSTEM !

\$95 → NO ROYALTIES ← \$95

(Mass. residents add 5% sales tax)

ORDERS OR INQUIRIES (800) 628-2828 ext. 712



ORDER NOW!

BEACON STREET SOFTWARE, INC.
P.O. BOX 216 • BEACON HILL • BOSTON, MA 02133



Circle no. 267 on reader service card.

Also available . . .

A Guide to Modula-2 K. Christian

1986/436 pp./46 illus./hardcover \$34.00
(Texts and Monographs in Computer Science)
ISBN: 0-387-96242-5

Ada® in Practice

C.N. Ausnit, N.H. Cohen, J.B. Goodenough, and R.S. Eanes
1985/216 pp./79 illus./softcover \$25.00
(Springer Books on Professional Computing)
ISBN: 0-387-96182-8

Pascal User Manual and Report

Revised for the ISO Pascal Standard
Third Edition

K. Jensen and N. Wirth

1985/266 pp./76 illus./softcover \$15.50
ISBN: 0-387-96048-1

To order, or for more information, please write to: Springer-Verlag New York, Inc., Attn: G. Kiely, 175 Fifth Avenue, New York, NY 10010



Springer-Verlag

New York Berlin Heidelberg
London Paris Tokyo

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

PolyBoost - Run 2 to 10 times faster with software accelerator. Speeds disk access, screen display, keyboard input. PC \$ 69

AI-Expert System Dev't

Arity System-incorporate with C programs, rule & inheritance MS \$ 259
Experteach - Powerful, no limit on memory size. Samples PC \$ 399
EXSYS MS \$ 339
 Texas Instruments:
 PC Easy PC \$ 439
 Personal Consultant PLUS PC \$ 2599

AI-Lisp

Microsoft MuLisp 85 MS \$ 199
 PC Scheme LISP - by TI PC \$ 85
TLC LISP - classes, compiler. MS \$ 225
 TransLISP - learn fast MS \$ 85
 Others: IQ LISP (\$155), UNX LISP (\$59), IQC LISP (\$269), WALTZLISP (\$139)

AI-Prolog

APT - Active Prolog Tutor - build applications interactively PC \$ 65
ARITY Standard - full, 4 Meg Interpreter - debug, C, ASM PC \$ 319
COMPILER/Interpreter-EXE PC \$ 699
 With Exp Sys, Screen - KIT PC \$ 1129
 LPA MacProlog Complete - incremental compiler and an interpreter MAC \$ 295
 LPA MicroProlog - intro MS \$ 85
 LPA MicroProlog Prof. full memory MS \$ 349
 Prolog-86 - Learn Fast, Standard, tutorials, samples MS \$ 89
 Prolog-86 Plus - Develop MS \$ 229
 TURBO PROLOG by Borland PC \$ 69

AI-Other

METHODS - SMALLTALK has objects, windows, PC \$ 69
 Q'NIAL - Combines APL with LISP. Source or binary. PC \$ 349
 Smalltalk/V-graphics PC \$ 89

Atari ST & Amiga

We carry full lines of Manx, Lattice, & Metacomco.
Amiga - LINT by Gimpel Amiga \$ 79
 Cambridge LISP Amiga \$ 200
 Lattice C ST, Amiga \$ 139
 Lattice Text Utilities Amiga \$ 75
 Megamax - tight, full ST \$ 200

FEATURES

dBXL by Word Tech - complete interpreter clone. Adds windowing. Quicksilver, LAN support. Non-copy protected. PC\$ 129

C86 PLUS - New version has 70% faster execution speed, tight code full ANSI library. Support for source level debugger. ROMable. Library C source. Ask about benchmarks. MS \$ 459

700+ Programmer's Products

The Programmer's Shop carries every programmer's software product for MSDOS, PCDOS, CPM, Macintosh, Atari, and Amiga systems. We help you choose the best tools for you. Most popular products are in stock, available for quick delivery. We will gladly special order a product for you at no charge — just allow a few extra days for delivery.

Need Cross Compilers, Translators, or the right Fortran compiler? Ask us.

- **Programmer's Referral List**
- **Compare Products**
- **Help find a Publisher**
- **Evaluation Literature FREE**
- **BBS - 7 PM to 7 AM 617-826-4086 National Accounts Center**
- Our Services:
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products

Basic

Basic Development System - for BASICA; Adds Renum, more. PC \$ 105
 Basic Development Tools by Sterling Castle PC \$ 89
 Basic Windows by Syscom PC \$ 95
 BetterBASIC - all RAM, modules Structure. Full BASICA PC \$ 139
 8087 Math Support PC \$ 89
 Run-time Module PC \$ 179
 Better Tools - for Better Basic PC \$ 95
 CADSAM FILE SYSTEM-full MS \$ 69
 GoodBas - maintain code PC \$ 95
 LPI Basic - MS compatible UNIX \$1100
 Prof. Basic - Interactive, debug PC \$ 79
 8087 Math Support PC \$ 47
 QuickBASIC V2.0-New interface PC \$ 69
 TRUE Basic - ANSI PC \$ 119
 Run-time Module PC \$ 459

Cobol

Macintosh COBOL - full MAC \$ 459
 MBP - Lev. II, native MS \$ 819
 Microsoft COBOL MS \$ 439
 Microsoft Cobol Tools - xref, debugger w/source support. Xenix \$319 PC \$ 209
 Professional COBOL PC \$ 2695
 Realia - very fast MS \$ 819
 Ryan McFarland COBOL MS \$ 699
 COBOL-8X MS \$ 995
 VS Workbench PC \$3500

Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure PC Call
 EMACS by UniPress - powerful, multifile, windows Source:\$929 \$ 299
 Epsilon - like EMACS, full C-like language for macros. PC \$ 155
 KEDIT - like XEDIT PC \$ 105
 Lattice Screen Editor - multiwindow, multitasking Amiga \$ 89 MS \$ 109
 PC/VI - Custom Software new version fast MS \$ 109
 Personal REXX PC \$ 109
 PMATE - power, multitask PC \$ 119
 SPF/PC - fast, virtual memory PC \$ 139
 XTC - multitasking PC \$ 79

C Libraries-Communications

Asynch by Blaise PC \$ 135
 Greenleaf Comm Lib. PC \$ 149
 Multi-Comm - add multitasking, use w/Multi-C PC \$ 149
 Software Horizons pack 3 PC \$ 119

RECENT DISCOVERY

Uniware Cross Development Tools include 68000 C compiler. Development Package with compiler, assembler, link editor, and utilities, 17 cross assemblers for Intel, TI, Motorola, Zilog, etc. - relocatable, macros. MS Call

C Language-Compilers

| | |
|--|----------|
| AZTEC C86 - Commercial | PC \$499 |
| C86 by CI - 8087, reliable | MS \$299 |
| Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev's Kit | PC \$ 77 |
| HOT C - new, intriguing | PC \$ 85 |
| Lattice C - from Lattice | MS \$299 |
| Mark Williams - w/debugger | MS \$369 |
| Microsoft C 4.0- Codeview | MS \$279 |
| Wizard C - Lattice C compatible, full sys. III, lint, fast. | MS \$359 |

C Language-Interpreters

| | |
|--|----------|
| C-terp by Gimpel - full K & R | MS \$229 |
| C Trainer - by Catalytix | PC \$ 89 |
| INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs | MS \$379 |
| Interactive C by IMPACC Assoc. Interpreter, editor, source, debug. | PC \$225 |
| Introducing C-self paced tutorial | PC \$105 |
| Run/C Professional | MS \$179 |
| Run/C Lite - improved | MS \$ 97 |

C Libraries-General

| | |
|--|----------|
| Blackstar C Function Library | PC \$ 79 |
| C Essentials - 200 functions | PC \$ 83 |
| C Food by Lattice-ask for source | MS \$109 |
| C Scientific Subroutines - Peerless | MS \$135 |
| C Tools Plus (1 & 2) | PC \$135 |
| C Utilities by Essential - Comprehensive screen graphics, strings, source. | PC \$137 |
| C Worthy Library - Complete, machine independent | MS \$269 |
| Entelekon C Function Library | PC \$119 |
| Entelekon Superfonts for C | PC \$ 45 |
| Greenleaf Functions-portable, ASM | MS \$139 |
| PForCe by Phoenix - objects | PC \$249 |

C Libraries-Files

| | |
|--|-----------|
| FILES: C Index by Trio - full B + Tree, vary length field, multi compiler /File is object only | MS \$ 89 |
| /Plus is full source | MS \$349 |
| CBTREE - Source, no royalties | MS \$ 99 |
| CTree by Faircom - no royalties | MS \$319 |
| dbQUERY - ad Loc, SQL - based | MS \$159 |
| dbVISTA - full indexing, plus optional record types, pointers, Network. | |
| Object only - MS C, LAT, | C86 \$159 |
| Source - Single user | MS \$429 |
| Source - Multiuser | MS \$849 |
| dBASE Tools for C | PC \$ 65 |
| dbc Isam by Lattice | MS \$199 |
| dBx - translator w/source | MS \$319 |
| | MS \$519 |

FEATURE

Insight 2+ - Flexible expert systems shell has intrinsic dBASE operators, forward and backward chaining, transparent view of reasoning process. MS \$389

Circle no. 133 on reader service card.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

C PROGRAMMER'S SUPPORT PRODUCTS

Professional tools shorten your development time, help you produce cleaner code, and write better user interfaces. Your programming problems have already been solved. Call our C specialist TODAY:

C Support-Systems

| | |
|---|----------------|
| Basic-C Library by C Source | PC \$139 |
| C Sharp - well supported. Source, realtime, tasks, state system | PC \$600 |
| C ToolSet - DIFF, xref, source | MS \$ 95 |
| The HAMMER by OES Systems | PC \$159 |
| Lattice Text Utilities | MS \$ 95 |
| Multi-C - multitasking | PC \$149 |
| PC LINT-Checker. Amiga | MS \$107 |
| SECURITY LIB-add encrypt to C, C86 programs. Source | \$229 PC \$115 |
| Quickshell - script compiler | PC \$349 |

C-Screens. Windows. Graphics

| | |
|--------------------------------|----------|
| C Power Windows by Entelekon | PC \$119 |
| dBASE Graphics for C | PC \$ 69 |
| Curses by Lattice | PC \$ 99 |
| ESSENTIAL GRAPHICS - fast | PC \$209 |
| GraphiC - mono version | PC \$217 |
| GraphiC - new color version | PC \$295 |
| Greenleaf Data Window w/source | PC \$199 |
| Multi-Windows - use w/ Multi-C | PC \$295 |
| Screen Ace Form Master | PC \$195 |
| Vitamin C - screen I/O | PC \$129 |
| Windows for C - fast | PC \$159 |
| Windows for Data - validation | PC \$239 |
| ZView - screen generator | MS \$189 |

Debuggers

| | |
|---|----------|
| 386 Debug | PC \$139 |
| Advanced Trace-86 by Morgan | |
| Modify ASM code on fly. | |
| CODESMITH - visual, modify and rewrite Assembler | PC \$125 |
| C SPRITE - data structures | PC \$107 |
| DSD87 - by Soft Advances | PC \$139 |
| Periscope I - own 16K | PC \$ 89 |
| Periscope II - symbolic, "Reset Box," 2 Screen | PC \$239 |
| Pfix-86 Plus Symbolic Debugger by Phoenix - windows | PC \$119 |
| Showcase by Test Software | PC \$239 |
| Software Source by Atron - Lattice, MS C, Pascal, Windows | PC \$135 |
| single step, 2 screen, log file. w/Breakswitch | MS \$115 |
| w/Breakswitch | \$199 |

FEATURE

| | |
|------------------------------------|-------|
| TransLISP PLUS - with C INTERFACE. | |
| 400 + COMMON LISP functions | |
| Optional UNLIMITED Runtime | \$150 |
| PLUS for MSDOS | \$179 |

Order before 2/28/87 and mention this ad for these special prices.

| | List | Normal | SPECIAL |
|--------------------------------|-------|--------|---------|
| cTree by Faircom | \$395 | \$319 | \$279 |
| Pre-C - (Lint-Like) | \$295 | \$199 | \$179 |
| Run C Professional | \$250 | \$179 | \$159 |
| Panel - screen management | \$295 | \$219 | \$189 |
| dbc - for dBASE II or III | \$250 | \$199 | \$169 |
| C Utility Library by Essential | \$185 | \$137 | \$119 |
| ZView - screen management | \$245 | \$189 | \$169 |

Fortran & Supporting

| | |
|--|-----------|
| ACS Time Series | MS \$419 |
| Forlib+ by Alpha - graphics and file routines, comm. | MS \$ 59 |
| MACFortran by Microsoft | MAC \$229 |
| MS Fortran link to C | MS \$209 |
| No Limit - Fortran Scientific | PC \$119 |
| RM/Fortran - enhanced "IBM Professional Fortran" | MS \$389 |
| Scientific Subroutines - Matrix | MS \$149 |
| Statistician by Alpha | MS \$249 |
| Strings and Things - register, shell | PC \$ 59 |

Multilanguage Support

| | |
|--|-----------------------|
| BTRIEVE ISAM | MS \$199 |
| BTRIEVE/N-multiuser | MS \$469 |
| CODESIFTER - Profiler. | MS \$ 99 |
| HALO Graphics - 115+ devices. | |
| Animation, engineering, business. | |
| Any MS language, Lattice, C86 | PC \$209 |
| Informix - by RDS | PC \$639 |
| Informix 4GL - application builder | PC \$799 |
| Opt Tech Sort - sort, merge | MS \$119 |
| PANEL - Create screen with editor, generates code. Full data validation, no royalties. | Xenix \$539, MS \$219 |
| PolyLibrarian by Polytron | MS \$ 85 |
| PVCS Version Control | MS \$329 |
| QMake by Quilt Co. | MS \$ 84 |
| Rtrieve - Xtrieve option | MS \$119 |
| Screen Sculptor - slick, thorough, fast, BASIC, PASCAL. | PC \$ 99 |
| Xtrieve - organize database | MS \$199 |
| ZAP Communications - VT 100, TEK 4010 emulation, file xfer. | PC \$ 89 |

Pascal and Supporting

| | |
|---|----------|
| ALICE - learn Pascal, Turbo compatible, interpreter | PC \$ 68 |
| Exec - Chain Programs | MS \$ 85 |
| MetaWINDOW - graphics toolkit | PC \$135 |
| Microsoft PASCAL - faster | MS \$189 |
| MICROTEC PASCAL - 5 memory models, "Iterators", 65 bit 8087 strings | MS \$665 |
| Pascal Pac with Tidy - formatter, utilities | PC \$ 69 |
| Pascal Tools - strings, screen | PC \$109 |
| Pascal Tools 2 - by Blaise | MS \$ 85 |
| Pascal 2 - tight, fast | MS \$329 |
| Pfas - Portable Isam | MS \$185 |
| TurboHALO - 150 routines, IBM EGA, Hercules, more | PC \$ 99 |
| USCD Pascal - native code | MS \$ 69 |

RECENT DISCOVERY

C Scientific Library - 300 + C functions with source, no royalties includes matrix and vector, eigen system analysis, statistics, probability, regression, graphics support, MUCH more. PC \$245

Other Languages

| | |
|---|------------------|
| APL*PLUS/PC | PC \$ 469 |
| Artek ADA Compiler - DOD standard minus multitasking | PC \$ 469 |
| CLIPPER-dBASE Compiler | MS \$ 429 |
| Correct FORTH - ROMable | PC \$ 80 |
| ED/ASM - 86 by Oliver | PC \$ 85 |
| Lattice RPG II Compiler | PC \$ 719 |
| MacASM - fast | MAC \$ 99 |
| MasterForth - Forth '83 | MAC or PC \$ 109 |
| Microsoft MASM - faster | MS \$ 98 |
| Modula 2 by Volition Systems | MS \$ 250 |
| Modula-2/86 Compiler by Logitech w/ 8087 (\$ 99), 512K (\$145). | PC \$ 62 |
| Pasm - by Phoenix | MS \$ 109 |
| SNOBOL4+ - great for strings | MS \$ 85 |
| Turbo Edit/ASM - by Speedware | PC \$ 85 |

Xenix-86 & Supporting

| | |
|--|--------|
| Basic - by Microsoft | \$ 239 |
| Cobol - by Microsoft | \$ 639 |
| Fortran or Pascal - by Microsoft | \$ 439 |
| MicroFocus Lev. II Compact COBOL\$ 795 | |
| Xenix Complete System | \$1049 |

Other Products

| | |
|--|-----------|
| 386 Assembler/Linker | PC \$ 459 |
| ASMLIB - 170+ routines | PC \$ 135 |
| BSW Make - like UNIX make | MS \$ 85 |
| Dan Bricklin's Demo Program | PC \$ 59 |
| dBrief - Customize BRIEF for dBASE development. with BRIEF \$275. | PC \$ 95 |
| H Test/H Format - XT Fix | PC \$ 89 |
| Help/Control - on line help | PC \$ 125 |
| Interactive Easyflow-HavenTree | PC \$ 129 |
| Link & Locate - tools to work with Intel and Tektronix projects. | MS \$ 329 |
| LMK - like UNIX make | MS \$ 149 |
| Microsoft Windows | PC \$ 69 |
| Software Development Kit | PC \$ 329 |
| MKS Toolkit - Unix, vi, awk | PC \$ 119 |
| PDisk - cache, tree | PC \$ 109 |
| PMaker - by Phoenix | PC \$ 79 |
| Polymake by Polytron | MS \$ 79 |
| PS MAKE by UniPress | MS \$ 79 |
| RPG II - by Lattice | PC \$ 719 |
| SECRET DISK by Lattice | PC \$ 95 |
| SET:SCIL - manager revisions | PC \$ 299 |
| Shrink/Shrinkem - put more files on disk with spacemaker | PC \$ 150 |
| SoftEst - Manage projects. | MS \$ 350 |
| Synergy-Create user interfaces | MS \$ 375 |
| Texsys - control source | MS \$ 89 |
| Tom Rettig's Library - dBASE | PC \$ 89 |
| Visible Computer: 8088 - Simulates demos or any .exe. com, Debugger. 350 pg. tutorial, unprotected | PC \$ 65 |

Note: All prices subject to change without notice.
Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add \$3/item.

Call for a catalog, literature, advice and service you can trust



HOURS



8:30 AM - 8:00 PM EST.

Circle no. 141 on reader service card.

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339

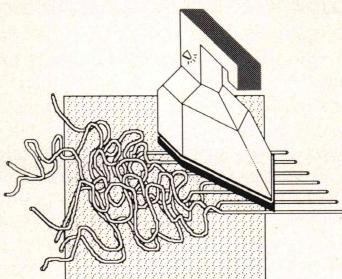
Mass: 800-442-8070 or 617-826-7531 12/86

"The scope and detail of services you provide are exemplary — it's obvious you have given a lot of thought to what information people need . . . For someone like myself, critical appraisals of software and comprehensive collections of offerings such as you have are really useful."

A. Bruce Cyr

Foundation of American College of Health Care Administrators

THE STATE OF BASIC



Modules in True BASIC

Two years ago, John Kemeny and Thomas Kurtz (the creators of BASIC) and a group of young enthusiastic entrepreneurs launched True BASIC, an implementation of BASIC that accesses up to 640K of memory and provides structured code constructs (some even out-muscle Pascal and Modula-2) while maintaining simple data types—giving programmers both numbers and strings! True BASIC dared to introduce structured code into a microcomputer BASIC dialect. A year later Microsoft launched its QuickBASIC compiler, which extends the syntax of BASIC in the same direction as that of True BASIC. The second version of QuickBASIC adds a few more extensions. In addition, both True BASIC and QuickBASIC support external libraries. Last November, Borland International announced Turbo BASIC, which, not surprisingly, also has structured code features. Then, in the same month, Kemeny and Kurtz released Version 2.0 of True BASIC, which supports modules and such features as the ability to load libraries and modules into memory.

We asked Dr. John Kemeny about the importance of modules in True BASIC. His answer was: "First, you have to talk about modules and work spaces together—I didn't realize, at first, the importance of work spaces. I think it's these two features together that make True BASIC 2.0 exciting for me. No matter how good a language is, there are always more features you would like to add. With modules and work spaces, True BASIC really becomes an extendable language. Implementing separately compiled libraries was step 1. Implementing modules was the next logi-

cal step. Using modules, the main program can be truly ignorant of what is happening within the library. Routines that used to take 20 parameters now take 3, and code seems more focused. Loading modules into a work space gives you the speed and immediacy that I always liked about BASIC. My functions, imported from modules and libraries, seem to be built-in—I don't need to declare them. I can concentrate on the code portion I am working on and take the rest for granted in this part of the environment."

True BASIC modules allow more sophisticated interaction of routines and data. We'll now describe briefly the components of a module and give a short example.

Modules enable you to declare three scope levels for variables. First, the *PUBLIC* declaration lists the names of all global variables that can be exported. These variables can be accessed by any routine within the module and by any program that uses that same module. In contrast, the *SHARE* declaration lists the variables that are accessible only by the routines within the module. Moreover, shared variables are static, enabling them to retain their values between calls to different module routines. Module developers can thus use shared variables to create and maintain data structures that are invisible to client programs. The third declaration, *PRIVATE*, lists variables that are local to the routines in which

```

MODULE Factorial
PUBLIC Last_Fact! Global variable
PRIVATE Bad_Number! Local variable
DECLARE DEF Bad_Number! Function declaration
SHARE Fact_Array(30), Product, MAX! Local static variables

!----- Initialize module -----
let MAX = 30
let Last_Fact = Bad_Number
let Product = Bad_Number
IF MAX > 30 THEN MAT REDIM Fact_Array(MAX)! Adjust array size
let Fact_Array(1) = 1
FOR I = 2 TO MAX
    let Fact_Array(I) = I * Fact_Array(I-1)
NEXT I

!----- LOCAL ROUTINES -----
DEF Bad_Number = -1.0E+200

!----- EXPORTED ROUTINES -----
DEF Fact(N)

IF (INT(N) - N) < > 0 THEN
    let Fact = Bad_Number
ELSE
    let Last_Fact = Product
    IF N <= MAX THEN
        let Product = Fact_Array(N)
    ELSE
        let Product = Fact_Array(MAX)
        FOR I = MAX+1 TO N
            let Product = I * Product
        NEXT I
    END IF
    let Fact = Product
END IF
END DEF

END MODULE

```

Code Example 1: Sample True BASIC module

they appear—routines listed in these declarations cannot be exported.

Module initialization is also available and can be used to assign values to variables, open file buffers, execute module routines, and so on. Modules are automatically initialized just before a client program starts running.

Code Example 1, page 142, shows a simple module, Factorial, which provides a routine to calculate factorials. It is customized such that the first 30 factorials (assumed to be in the range most frequently used) are stored in a local static array. A factorial of up to 30 is simply recalled from memory, and factorials for higher numbers are calculated. The *PUBLIC* variable *Last_Fact* returns the last valid factorial obtained. It is accessible to the client program. The *PRIVATE* function *Bad_Number* is local to the module and works by returning a large negative number if the factorial of a non-integer is requested. The *SHARE* declaration lists the array *Fact_Array*, which stores the first 30 most frequently used factorials. The scalar *MAX* stores the upper limit of the factorial array. The values of the array are assigned during the module initialization phase. During that same phase, the array may be expanded if necessary. You can easily customize the program by assigning a value other than 30 to *MAX*. Notice that function *Fact* calculates the factorials of numbers greater than 30, starting with *Fact_Array*(*MAX*).

BASIC is back on its feet. The language written off and belittled by many is making a comeback not to be taken lightly.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 10.

The Advanced Programmer's Editor That Doesn't Waste Your Time

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Multiple windows, files
- Unlimited file size, line length
- 30 day money-back guarantee
- Great on-line help system
- Regular Expression search
- Supports large displays
- Not copy protected

Only \$195
Lugaru
Software Ltd.

5740 Darlington Road
Pittsburgh, PA 15217

Call
(412) 421-5911

for IBM PC/XT/AT's or compatibles

Circle no. 135 on reader service card.

The C Programmer's Assistant

C TOOLSET



UNIX-like Utilities for Managing C Source Code

No C programmer should be without their assistant—CToolSet. All of the utility programs are tailored to support the C language, but you can modify them to work with other languages too.

Source code in standard K&R C is included; and you are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

12 Time Savers

DIFF - Compares text files on a line-by-line basis; use *CMP* for byte-by-byte. Indispensable for showing changes among versions of a program under development.

GREP - Regular expression search. Ideal for finding a procedural call or a variable definition amid a large number of header and source files.

FCHART - Traces the flow of control between the large modules of a program.

PP (C Beautifier) - Formats C program files so they are easier to read.

CUTIL - A general purpose file filter.

CCREF - Cross references variables used within a program.

CBC (curly brace checker) - checks for pairing of curly braces, parens, quotes, and comments. Other utilities include *DOCMAKE*, *ASCII*, *NOCOM*, and *PRNT*.

Source code to every program is included!

Thorough User Support Text and Online

C ToolSet documentation contains descriptions of each program, a listing of program options (if any), and a sample run of the program.

On-line help gives you information on the programs and how to run them. Most of the programs respond to -? on the command line with a list of options.

Call **800-821-2492** to order C ToolSet risk-free for only \$95.

**Solution
Systems™**

335-D Washington St.,
Norwell, MA 02061
(617) 659-1571

Full refund in 30 days..

Circle no. 152 on reader service card.

Professional Programming Products

for Microsoft C, PASCAL, FORTRAN, and Assembly Language



FREE

PC-WRITE™ text editor, and
SOURCE CODE



**PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!
NO ROYALTIES REQUIRED**

ASMLIB

The Programmer's Library

- A Multipurpose set of over 200 Assembly Language sub routines supplied in the form of a linkable library.
- Virtual disk file handling.
- Int. driven asynch. support.
- Graphics on EGA, herc. and CGA.
- Floating point math and trig routines with 8087 support.
- Installable keyboard activated programs are easily written with ASMLIB's special functions.
- Plus much, much more.
- Supplied with complete source code.

Only \$149.00 Complete

asmTREE

The Programmer's B+Tree Data File Management System

- A complete single/ multiuser database management system written entirely in Assembly Language gives the Lattice "C" or Assembly Language programmer these capabilities.
- Up to 256 users.
- Up to 256 index and data files.
- Multiple key types.
- Multiple indices per index file.
- Duplicate and variable length keys.
- Virtual file handling
- Plus much, much more
- Supplied with complete source code.

Only \$395.00 Complete

NEW

NET-TOOLS - Network Programming Tools

NET-TOOLS allows you to write programs for ANY NETBIOS compatible local area network - fast and easily.

- ★A multitude of subroutines allow your program to handle all network tasks directly.
- ★Redirect Local Devices simply and easily with a single function call.
- ★Send and Receive disk files with error detection. Issue a single call to NET-TOOLS, and it will automatically send a file of any length to a NET-TOOLS receiver.
- ★Send and Receive Messages with automatic retries and error detection. Both the datagram and session protocols are available to your program.

★COMPLETE SOURCE CODE IS PROVIDED - written in assembly language.

ONLY \$149.00 Complete

B C ASSOCIATES

3261 No. Harbor Blvd., Suite B
Fullerton, CA 92635

1-800-262-8010

in Calif. Call

(714) 526-5151

*FREE Assembly Language SOURCE CODE !

Outside CA, call TOLL FREE 1-800-262-8010

USE YOUR VISA OR MASTERCHARGE -



All prices include UPS shipping within continental United States. Outside U.S. please add \$10 per package. Calif. residents please add 6.5% sales tax.

Circle no. 182 on reader service card.

IF YOU NEED \$5,000...\$20,000 EVEN UP TO \$500,000 TO START A NEW BUSINESS OR TO EXPAND AN EXISTING FIRM—THEN READ WHY YOU TOO WILL CALL THIS INCREDIBLE MONEY RAISING

BUSINESS OPPORTUNITY SEEKERS' LOANS MANUAL “The Small Business Borrower’s Bible”

Practically prepares the loan application for you line-by-line...the “proper” way.
All properly prepared applications are processed faster...no red tape!

EVEN
LOAN DOLLAR
YOU GET
YOU KEEP
AND USE TO
OPERATE
YOUR BUSINESS

Guaranteed Loans...Direct Loans...and Immediate Loans are available now!

Most men and women seriously interested in starting their own business are eligible to apply — including those who already own a business and need capital fast for expansion...or to stay afloat...even if they've been flatly refused by banks and turned down elsewhere! Yet, too many never qualify, simply because they do not know how to “properly” prepare the loan application...

In order to help those people applying for these guaranteed and direct loans fill out their loan applications the “right way” our business research along with diligent compilation and effective efforts, has successfully assembled and published a comprehensive, easy-to-follow seminar manual: The Business Opportunity Seekers’ Loans Manual, that will quickly show you practically everything you’ll need to know to prepare a loan application to get federally Guaranteed and Direct Loans.

Here are just some of the many important benefits the Business Opportunity Seekers’ Loans Manual provides you with:

- a completely filled in sample set of actual SBA loan application forms, all properly filled in for you to easily follow—aids you in quickly preparing your own loan application the right way. Each line on the sample application forms is explained and illustrated in easy-to-understand language.
- fast application preparation procedures for getting loans for both new start up business ventures and established firms.
- advises you on how to properly answer key questions necessary for loan approval and in order to help avoid having your application turned down—gives you advice on what you should not do under any circumstances.
- what simple steps you take to guarantee eligibility—no matter if you do not presently qualify.
- where you can file your application for fastest processing.

At this point the most important question you want answered is: Just where is all this loan money coming from? Incredible as it may sound—these Guaranteed Loans, Direct Loans...and Immediate Loans are indeed available right now — from the best, and yet, the most overlooked and frequently the most ignored and sometimes outright ridiculed...“made-fun-of” source of ready money fast capital, in America — THE UNITED STATES GOVERNMENT

Of course, there are those who upon hearing the words “UNITED STATES GOVERNMENT” will instantly freeze up and frown and say

“only minorities can get small business loan money from the government!”

Yet, on the other hand (and most puzzling) others will rant on and on and on that

“don’t even try, it’s just impossible — all those Business Loans Programs are strictly for the Chryslers, the Lockheeds, the big corporations, not for the little guy or small companies” etc.

GUARANTEED YOUR LOAN MUST BE APPROVED...OR MONEY BACK — ONLY A SMALL PRICE TO PAY FOR THE LOAN YOU CAN GET...NO RISK AND NO HASSLES.

BUSINESS OPPORTUNITY SEEKERS' LOANS MANUAL

Still there are those who declare:

“...I need money right now...and small business government loans take too darn long. It’s impossible to qualify. No one ever gets one of those loans.”

Or you may hear these comments:

“...My accountant’s junior assistant says he thinks it might be a waste of my time!” “Heck, there’s too much worrisome paperwork and red tape to wade through!”

Frankly — such rantings and ravings are just a lot of “bull” without any real basis — and only serve to clearly show that lack of knowledge...misinformation...and not quite fully understanding the UNITED STATES GOVERNMENT’S Small Business Administration’s (SBA) Programs have unfortunately caused a lot of people to ignore what is without a doubt — not only the most important and generous source of financing for new business start ups and existing business expansions in this country — but of the entire world!

Now that you’ve heard the “bull” about the United States Government’s SBA Loan Program — take a few more moments and read the following facts:

- Only 9.6% of approved loans were actually made to minorities last year
- What SBA recognizes as a “small business” actually applies to 97% of all the companies in the nation
- Red tape comes about only when the loan application is sent back due to applicant not providing the requested information...or providing the wrong information
- The SBA is required by Congress to provide a minimum dollar amount in business loans each fiscal year in order to lawfully comply with strict quotas. (Almost 5 billion this year)

Yet, despite the millions who miss out — there are still literally thousands of ambitious men and women nationwide who are properly applying — being approved — and obtaining sufficient funds to either start a new business, a franchise, or buy out or expand an existing one. Mostly, they are all just typical Americans with no fancy titles, who used essentially the same effective know-how to fill out their applications that you’ll find in the Business Opportunity Seekers’ Loans Manual.

So don’t you dare be shy about applying for and accepting these guaranteed and direct government loans. Curiously enough, the government is actually very much

GUARANTEE #1

Simply — look over this most effective money raising loan preparation assistance manual for 15 days — and, then, if you are not convinced that it can actually help you obtain the Business Loan you need right away — just return it for a full refund and prompt refund.

GUARANTEE #2

Even after 15 days — here’s how you are still strongly protected — if you decide to keep the manual — and you apply for an SBA Loan anytime within 1 year — your loan must be approved and you must actually receive the funds or your money will be refunded in full.

Only because we are so confident that this is a fact do we dare make such a strong binding seldom-heard-of Double Guarantee. No stronger guarantee possible!

Of course, no one can guarantee that every request will be approved—but clearly we are firmly convinced that any sound business request properly prepared—showing a reasonable chance of repayment and submitted to SBA—will be approved.

THOUSANDS ARE PROPERLY APPLYING AND BEING APPROVED. HERE'S YOUR CHANCE TO JOIN THEM!

FREE BONUS

If you order your manual today you’ll receive a valuable treasury of fast, easy, low-capital and highly profitable business programs worth forty-five dollars — yours absolutely free!

100% tax deductible as a business expense. Don’t delay — order your copy today!

NO RISK LOAN OPPORTUNITY FORM

Detach and rush for
COMPLETE PREPARATION ASSISTANCE FOR LOAN APPROVAL

Please rush me _____ copies of “Business Opportunity Seekers’ Loans Manual” each at a \$20 fee plus \$3.00 handling and shipping. I am fully protected by the two strong guarantees above. I’m ordering today — so I can receive FREE—the valuable treasury of fast, easy, low-capital and highly profitable business programs ... worth forty-five dollars — mine free to keep even if I decide to return the manual for a full refund.

Enclosed is Full Payment
 Cash Check Money Order
Send payment with order.

Name _____
Please Print Clearly
Address _____

City _____

State _____ Zip _____

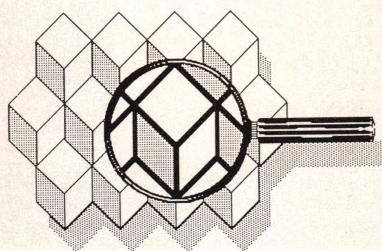
MAIL TO:

Financial Freedom Publishers
110 W. 5th St. Dept. ST-1
Winston-Salem, NC 27101

ORDER NOW

©1985

OF INTEREST



Languages

A new Modula-2 compiler for PC-DOS and MS-DOS is available from **Farbware**. The product is a complete native code compiler, code generator, and run-time package. It implements the full Modula-2 language, including REALS and DECIMALS, and a comprehensive PC-DOS system interface. The unprotected program sells for \$89.95. Reader Service No. 16.

Farbware
1329 Gregory
Wilmette, IL 60091
(312) 251-5310

A PROLOG for the Macintosh has been announced by **Advanced A.I. Systems**. AIIS PROLOG provides efficient methods of matching (or unification) and rule retrieval and allows each section of code to be tested as it's keyed in. Other features include Edinburgh syntax; compatibility with DEC 10/20 PROLOG, C-PROLOG, and Quintus PROLOG; a package system for multiple memory-based databases; and a PROLOG pretty printer. Debugging facilities include an extensive interactive debugger, illegal-argument checking, and unknown-function checking. AIIS PROLOG requires a Macintosh with at least 512K RAM and costs \$150. Reader Service No. 17.

Advanced A.I. Systems Inc.
P.O. Box 39-0360
Mountain View, CA 94039
(415) 961-1121

Version 2.0 of **Kyan Software**'s Pascal for the Apple is now available. This version is a fully validated implementation of ISO Pascal and runs on any Apple II with 64K RAM. Kyan Pascal includes a full-screen text editor, a

native code compiler, a macro-assembler, utilities, and several Pascal extensions. The built-in macro assembler allows programmers to add inline assembly-language source code to their Pascal programs. The unprotected software is priced at \$69.95. An advanced version of the software called Kyan Pascal Plus is priced at \$99.95. Reader Service No. 18.

Kyan Software
1850 Union St., #183
San Francisco, CA 94123
(415) 626-2080

Computer Innovations has released a new C compiler, C86PLUS, which is based on a technology that applies artificial-intelligence techniques to produce highly optimized code. C86PLUS takes advantage of newer, more powerful hardware architectures, such as Intel's 80286 and 80386 microprocessors. The program runs the Sieve benchmark 20 percent faster than does Microsoft C, Version 4.0, and is 70 percent faster than the current C86, Version 2.3. C86PLUS runs on the IBM PC, PC/XT, PC/AT, and compatibles and sells for \$497. Reader Service No. 19.

Computer Innovations
980 Shrewsbury Ave.
Tinton Falls, NJ 07724
(201) 542-5920

The Software Factory has introduced OPAL, a full-function interpretive language for PC application developers. OPAL integrates itself with DOS and applications running under DOS and offers a high degree of programming capabilities. OPAL sells for \$169. Reader Service No. 20.

The Software Factory Inc.
15301 Dallas Pkwy., Ste. 750 LB 44
Dallas, TX 75248
(214) 490-0835

CET Technology has released CET BASIC, a compiled application development language for Intel-based Unix and Xenix systems. The CET BASIC compiler is compatible with OA-SIS, THEOS, and UX-BASIC. CET BASIC programs can be intermixed with programs and subroutines in other languages. Additional features include multiuser support for ISAM, direct and sequential files, terminal in-

dependence, error trapping, program chaining, and COBOL-like formatting. The CET BASIC Compiler sells for \$695. Reader Service No. 21.

CET Technology Inc.
5405 Garden Grove Blvd., Ste. 160
Westminster, CA 92683
(714) 895-4345

PL/PC, a new programming language from **Creative Computer Software**, is based on APL with Modula-2 control structures. It offers an integrated, interactive programming environment and a spreadsheet-like data editor. Structured programming is supported, and English keywords are used instead of APL symbols. Debugging facilities include tracing, stopping, single-stepping, timing, and profiling. PL/PC requires an IBM PC or compatible with at least 360K RAM and DOS 2.11 or later. A demo version is available for \$16, a standard version for \$89, and an 8087 version for \$159. Reader Service No. 22.

Creative Computer Software
117 York St.
Sydney, NSW 200
Australia
(02) 261-1611

Turbo BASIC from **Borland International** combines the interactive aspects of BASIC with the structured, modular approach of Pascal. Turbo BASIC employs the same development environment as that of Turbo Pascal, including a memory-to-memory compiler, a full-screen editor, an internal linker and run-time library, and the Microcalc spreadsheet complete with source code. Turbo BASIC supports true recursion, full 8087 integration, and block-structured programming statements. The package runs on the IBM PC and compatibles and sells for \$99.95. Reader Service No. 23.

Borland International
4585 Scotts Valley Dr.
Scotts Valley, CA 95066
(408) 438-8400

DDJ

Why Are So Many People Switching to Smalltalk/V?

Why are scientists, engineers, and professionals switching to Smalltalk/V? Because it lets them do amazing things on their PCs, with a Mac-like interface and an easy-to-use object-oriented language. And with Smalltalk/V you get an unsurpassed array of problem-solving tools. You can even personalize the entire system to suit your needs.

Smalltalk/V is the programmable environment that gives you total control of your computer and makes it what it was meant to be, a truly personal tool for your mind.

"This is the real thing, folks. A super Smalltalk like this turns your PC into a hot workstation. It's fantastic . . . Highly recommended."

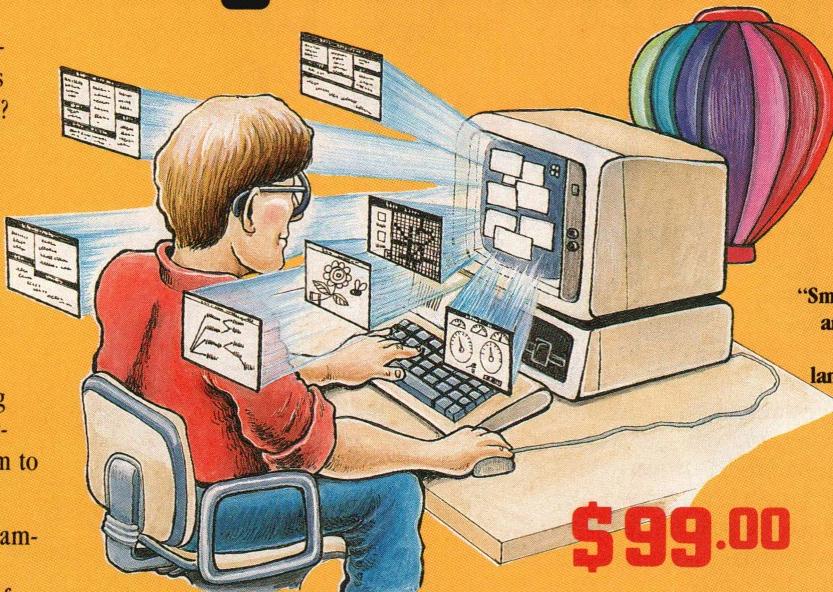
John C. Dvorak,
Contributing Editor, PC Magazine

"My background is in physical chemistry, not in programming. I like Smalltalk/V because I can use objects in the computer to represent objects in the physical world."

Dr. Paul Soper, Senior Specialist
E. I. du Pont de Nemours & Co.

"Smalltalk/V is a productive programming environment that allows us to quickly develop sophisticated medical applications."

Dr. Mike McCoy,
Dean for Instructional Computing
UCLA School of Medicine



\$99.00

Smalltalk/V

The Programmable Environment

"Smalltalk/V, with its visual interface and class structure, is a perfect way to simulate the complex interactions of natural systems."

Lee A. Graham, Research Assistant
Institute of Ecology, University of Georgia

"I solve problems quickly using Smalltalk/V because its classes and objects help me organize my thinking. And besides, it's fun to use."

Dr. Barry Fishman, Sr. Project Engineer
Hughes Aircraft Company

BYTE and BIX are trademarks of McGraw-Hill, Inc. IBM, IBM-PC, and IBM PC-AT are trademarks of International Business Machines Corporation. Unix is a trademark of Bell Laboratories.

"Smalltalk/V is the highest performance object-oriented programming system available for PCs."

Dr. Piero Scaruffi,
Chief Scientist, Olivetti
Artificial Intelligence Center

"Smalltalk/V is an excellent buy and makes a good alternative to other programming languages for the development of complex applications."

Bill Wong, Director, PC Labs
PC Magazine

Other Smalltalk/V Features

- Object-oriented Prolog integrated with the Smalltalk environment
- Supports exploratory programming and prototyping
- Class hierarchy with inheritance creates highly re-useable source code
- Smalltalk source code included, with browser windows for easy access and modification
- Object-swapping creates a virtual memory on hard or RAM disk
- Bit-mapped graphics with bit and form editors
- A sophisticated source-level debugger
- Automatic change log for easy recovery from errors
- Powerful directory/file browser system for organizing DOS files
- Access to other languages and DOS functions
- 500 page manual with comprehensive tutorial and reference sections
- Optional add-on modules
 - RS-232 communications interface with UNIX™ and TTY windows
 - EGA color graphics
 - "Goodies" diskette, including multiprocessing, music, zoom, object loader, and more

Yes! I want to turn my PC into a hot workstation! Send me . . .

| | |
|---|---------------|
| <input type="checkbox"/> Smalltalk/V - The Programmable Environment | \$99 |
| <input type="checkbox"/> Communications Option | \$49 |
| <input type="checkbox"/> EGA Color Option | \$49 |
| <input type="checkbox"/> "Goodies" diskette | \$49 |
| Shipping and Handling | \$ |
| CA residents add applicable sales tax | \$ |
| TOTAL | \$ |
| Shipping and Handling | \$ |
| U.S., Canada, Mexico | \$ 5.00 |
| Elsewhere | \$15.00 |

I enclose Check Money Order
 Credit card information MC VISA

Number: _____

Expiration: _____

Signature: _____

Name: _____

Street Address: _____

City/State/Zip: _____

Phone: _____

NOT COPY PROTECTED, 60-DAY MONEY-BACK GUARANTEE
ON-LINE USER-SUPPORT CONFERENCE ON BYTE'S BIX™

Smalltalk/V requires DOS and 512K RAM on IBM PCs (including AT) or "compatibles," and a CGA, EGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended.

digitalk inc.

5200 West Century Boulevard
Los Angeles, CA 90045 (213) 645-1082

How a magical Genie saved this programmer from going crazy

File format changes were driving me nuts. Then, I got stuck with a 12 year old word processing file that had to be converted to WordStar® ASAP. I must have passed out momentarily, because the next thing I remember is a guy in a turban, jamming a diskette into my PC. He pressed a few keys and "Presto," he created a new utility for my program. The file conversion underway, I passed out again and when I woke up, he was gone—but the amazing utility software he had used remained.

File Genie, that's the name on the diskette. An amazing tool. Since then, I discovered that it will convert word processing and data files from most any format to any other. And, as if that isn't enough, it also: diagnoses and fixes errors in broken database files, instantly searches and replaces on seven and eight bit data (on ambiguous file names, with the most complete set of regular expressions I've ever seen), has a script language with IF, ELSE, WHILE, FULL SCREEN CONTROL, and the ability to run DOS commands or programs, comes with on-line context sensitive help, et cetera, et cetera. Allows printing of files without my printer reacting to control codes too.

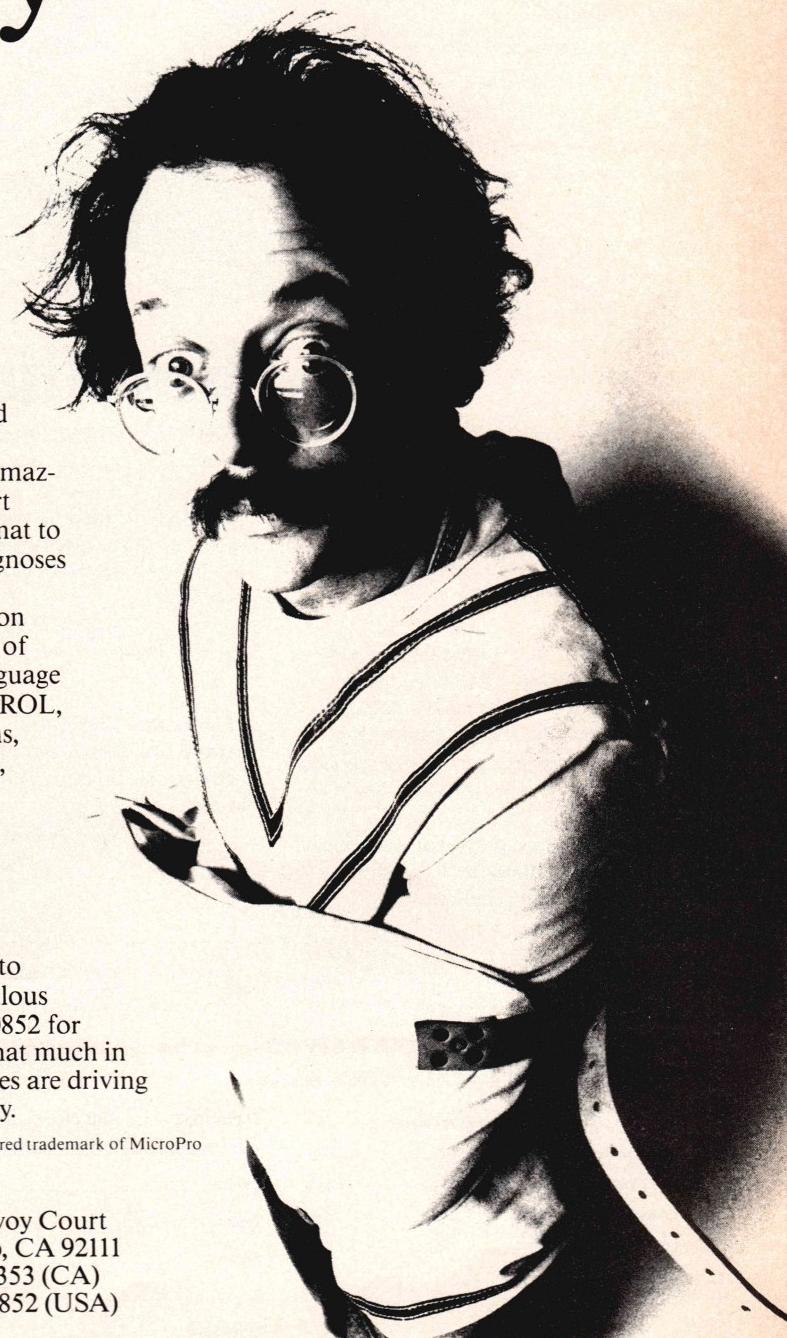
With **File Genie** I write my own utility programs quickly and easily. As you can imagine, I guarded this little gem with my life, keeping it totally secret. Until everybody kept asking me how I was doing all these incredible things. I have to admit, I am sold. Which gets me to the really fabulous news. **File Genie** is available by calling 1-800-822-0852 for an introductory price of only \$69.95. You'll save that much in aggravation the first time you use it. So if those files are driving you crazy, too, call and order your **File Genie** today.

WordStar® is a registered trademark of MicroPro



A software product of Team Austin Inc.

6809 Convoy Court
San Diego, CA 92111
619-278-5353 (CA)
800-822-0852 (USA)





to

**the dBx™ translator**

- dBx produces quality C direct from dBASE II or III programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- With power guidebook of conversion hints.
- Includes full screen handler and uses your current C database manager.
- May be used to move existing programs or help dBASE programmers learn C easily.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA. (Uses ANSI.SYS driver on MSDOS, CURSES under UNIX)
- Priced from \$350, also available from distributors.

dBx is a trademark of Desktop Ai1720 Post Road E., Westport, CT 06880 MCIMAIL • DESKTOPAI
Phone • 203-255-3400 Telex • 6502972226MCI

Circle no. 258 on reader service card.

EXIM ToolKit**BASIC Programming Support from EXIM Services****EXIM Services announces the EXIM ToolKit. A growing Library of over 65 Assembler and BASIC Programming Modules:**

- | | |
|---------------------------|-----------------------|
| ■ Video/Screen Management | ■ I/O File Management |
| ■ DOS Environment Support | ■ Date Arithmetic |
| ■ Data BASE Management | ■ General Purpose |

Increase Productivity, Decrease Development Time and Add Functionality to Your Applications.

A must addition for software developers using Microsoft QuickBASIC or IBM compilers. This high quality, low cost library offers developers the advantages of power and sophistication.

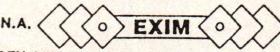
Window or Frame Screens and Messages . . . Save/Restore Full or Partial Screens . . . Accelerated Screen Displays . . . Horizontal/Vertical Scroll-Full or Partial Screen . . . Display "Pop Up" Help Screens . . . Find the First/Next Directory Entry . . . Sorting . . . etc.

Easy to use, high quality, the EXIM ToolKit is being offered for only \$49.95, plus \$5.00 shipping and handling, with a full money back guarantee.

"If you are not completely satisfied, return the diskettes and documentation within 30 days of purchase for a complete refund."

Do not be left behind. Bring your applications up to date with this NO RISK offer.

No licensing or royalty fees are required to distribute applications using the EXIM ToolKit.

EXIM SERVICES OF N.A.
P.O. BOX 5417
CLINTON, NEW JERSEY 08809

(201) 735-7640

Circle no. 371 on reader service card.

**Two great reasons to buy Turbo Pascal:
System Builder™ \$99⁹⁵ and Report Builder™ \$75⁰⁰****From the Designer Series by Royal American Technologies.**

State-Of-The-Art Program Generators that automatically build a **Relational Database** system without coding. Entry level "coders" can produce Database systems without coding. Developers have more flexibility and horsepower than any development tool on the market.

Self-documenting program includes screen schematics. System Builder will generate 2,000 lines of program code in approximately 6 seconds.

**SYSTEM BUILDER FEATURES:**

- Automatically generates Indented, Structured, Copy Book Source Code ready for compiling with Turbo Pascal (no programming needed)
- Paint Application and Menu screens using Keyboard
- Screens all use In-Line machine code for exceptional speed
- 16 Datafiles and 16 Index Keys per application
- Paint functions include: —Center, copy, move, delete, insert or restore a line with one keystroke
- Cut and paste blocks of text screen to screen
- Draw and erase boxes
- Access special graphic characters and character fill
- Go straight from screen to screen
- Define colors and intensities
- Support an unlimited number of memory variables
- File Recovery Program
- automatically modify existing datafiles
- Experienced developers can modify the System Builder
- Develop systems for Floppy or Hard Disk
- Modify System Builder's output source code to include External Procedures, Functions and Inline Code
- Easy-to-use Interface Program included to access ASCII and Dbase Files

REPORT BUILDER FEATURES:

- Automatically generates Indented, Structured Source Code ready for compiling Turbo Pascal (no programming needed)
- Automatically inter-

"I think it's wonderful . . . prospective buyers should seriously consider DESIGNER even before dBASE III."

Mr. Greg Weale
Corporate Accounts Manager,
Computerland

"We used DESIGNER last year to program a major application. It saved our programmers so much time. We now use DESIGNER instead of dBASE III as our development standard."

Mr. Peter Barge, Director
Services Division, Horwath & Horwath

Royal American Technologies
201 Sansome, Suite 202
San Francisco, CA 94104

(800) 654-7766in California (800) 851-2555
Ask for Operator 102.

Please rush me: _____ copies of SYSTEM BUILDER at \$99.95 per copy; _____ copies of REPORT BUILDER at \$75.00 per copy. I've enclosed \$5.00 for postage and handling. California residents add 6% sales tax.

Name _____

Address _____

City _____

State _____ Zip _____

Phone _____

Payment: Check Money Order Cashiers Check AMEX VISA MASTERCARD

Expiration date _____

Card Number _____

Signature _____

30-Day Money-Back Guarantee. Not copy-protected. \$10 restocking fee if envelope is opened. System Requirements: IBM PC/XT/AT[®], or similar, with minimum 256K RAM, dual floppy drives, or hard disk, color or monochrome monitor, MS-DOS[®] version 2.0 or later, Turbo Pascal Version 2.0 or later (Normal, BCD or 8087 versions).

[®]Trademark of International Business Machines Corp.

Turbo Pascal is a registered trademark of Borland International.

dBASE is a registered trademark of Ashton-Tate.

SYSTEM BUILDER PERFORMANCE
(Typical 10 screen 8 file/index application)**TASK**

- Planning and Design
- Screen Painting
- Programming
- Elapsed time to completed system

SYSTEM BUILDER DBASE III™

| | |
|-----------------------|------------|
| 60 minutes | 60 minutes |
| 15 minutes | 3 hours |
| 2 minutes | 10 hours |
| 1 hour and 17 minutes | 14 hours |

VARS, System Integrators and Dealers, let's work together. Head office: (415)397-7500**ROYAL AMERICAN**
Technologies Corporation™

Circle no. 312 on reader service card.

The Peak of Performance



SCALE THE HEIGHTS OF PRODUCTIVITY

Sure, you've proven that in your hands a computer is a productive tool. But if you haven't teamed up with a SemiDisk you have heights yet to climb!

IT'S NO MERE RAMDISK

SemiDisk has been leading the way for Disk Emulators since their inception. If you've seen RAMdisks you know what it's like to load programs in an

instant, and read or write files without delay. Unlike alternatives, the SemiDisk offers up to 8 megabytes of instant-access storage while leaving your computer's main memory free for what it does best - computing!

KEEP A GRIP ON DATA

Go ahead, turn off your computer. Take a vacation. With the battery backup option, your valuable data will be there in the morning even if you aren't. You'll sleep better knowing not even a 5 hour blackout will sabotage your files.

NEW LOWER SEMIDISK PRICES THAT WON'T SNOW YOU UNDER

| | 512K | 2Mbyte |
|--------------------|-------|--------|
| IBM PC, XT, AT | \$495 | \$995 |
| Epson QX-10 | \$595 | \$995 |
| S-100, SemiDisk II | \$799 | \$1295 |
| S-100, SemiDisk I | \$595 | — |
| TRS-80 II, 12, 16 | \$695 | \$1295 |
| Battery | | |
| Backup Unit | \$130 | \$130 |

Software drivers available for CP/M 80, MS-DOS, ZDOS, TurboDOS, and VALDOCS 2.

SEMDISK

SemiDisk Systems, Inc.

P.O. Box GG, Beaverton, Oregon 97075

503-626-3104

Call 503-646-5510 for CBBS/NW, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards. (300/1200/2400 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle no. 85 on reader service card.

ADVERTISER INDEX

| Reader Service No. | Advertiser | Page | Reader Service No. | Advertiser | Page | |
|--------------------|------------------------------|---------|--------------------|----------------------------------|-----------------------------|----|
| 369 | Aker Corporation | 22 | 220 | Nantucket | C2 | |
| 350 | Aldebaran Laboratories | 106 | 331 | Nirronics | 48 | |
| 321 | Alpha Computer Service | 53 | 342 | Oakland Group, Inc. | 46 | |
| 277 | Aspen Systems | 47 | 254 | Oasys | 31 | |
| 250 | Austin Code Works | 47 | * | On Command | 69 | |
| 182 | BC Associates | 144 | 357 | Oregon Software | 19 | |
| 267 | Beacon Street Software, Inc. | 139 | 214 | Periscope Co. Inc. | 88 | |
| 202 | Berkeley Decisions/Systems | 125 | 343 | Pharlap | 137 | |
| 159 | Blaise Computing | 2 | 229 | Port - A - Soft | 82 | |
| 217 | Blaise Computing | 74 | 129 | Programmer's Connection | 75 | |
| 161 | Borland International | C-4 | 129 | Programmer's Connection | 76-77 | |
| 212 | Burton Systems Software | 82 | 367 | Programmers Journal | 83 | |
| 235 | C Software Toolset | 44 | 334 | Programmer's Paradise | 59 | |
| 181 | C Users Group | 82 | 133,141 | Programmer's Shop | 140-141 | |
| * | C Ware | 98 | 301-304, | Programmer's Shop | 127 | |
| 307 | Cauzin Systems | 92-93 | 356 | Qualstar Corporation | 104 | |
| 370 | Code Blue | 42-43 | 144 | Quantum Computing | 117 | |
| 122 | Compu View | 49 | 107 | Quilt Computing | 48 | |
| 237 | CompuServe | 71 | 206 | Raima Corporation | 95 | |
| * | Creative Programming | 33 | 312 | Royal American Technologies | 149 | |
| * | CSSL, Inc. | 54 | * | SAS Institute | 131 | |
| 268 | Custom Software Systems | 137 | 168 | Sapiens Software | 78 | |
| 203 | DataLight | 9 | 210 | Scientific Endeavors | 104 | |
| 258 | Desktop A.I. | 149 | 114 | Seidl Computer Engineering | 84 | |
| 127 | Digitalink | 147 | 372 | Semi-Disk Systems | 150 | |
| * | Dr Dobb's Toolbook Shelf | 122-123 | 78 | SLR Systems | 44 | |
| 89 | Ecosoft, Inc. | 51 | 349 | SofCap, Inc. | 98 | |
| 90 | Edward K. Ream | 97 | 259 | Softfocus | 126 | |
| 138 | Essential Software | 13 | 372 | SoftWay | 112 | |
| 139 | Essential Software | 15 | 361 | Software Factory | 133 | |
| 371 | Exim | 149 | 314 | Software Garden Inc. | 56 | |
| 93 | Fair-Corn | 41 | 347 | Software Masters | 151 | |
| * | Financial Freedom Publishers | 145 | 168 | Software Security, Inc. | 73 | |
| * | Gimpel Software | 87 | 236 | Solution Systems | 27 | |
| 291 | Gold Hill Computers, Inc. | 1 | 287 | Solution Systems | 109 | |
| 97 | Greenleaf Software | 121 | 363 | Solution Systems | 143 | |
| 351 | Guidelines Software | 29 | 170 | Spencer Organization | 113 | |
| 132 | Harvard Softworks | 58 | 228 | Spencer Organization | 80 | |
| 280 | Hersey Micro Consulting | 60 | 345 | Springer-Verlag | 139 | |
| 327 | Integral Quality, Inc. | 21 | 204 | STAT Toolbox for Turbo Pascal | 63 | |
| 179 | Intel Corporation | 38-39 | 364 | Stonybrook Software | 72 | |
| 355 | Jou Laboratories | 61 | 236 | Summit Software Technology, Inc. | 129 | |
| 294 | Kurtzberg Computer Systems | 86 | 320 | Symmetric Computer Systems | 32 | |
| 266 | Language Processors, Inc. | 99 | 148 | T.O.C Business Solutions | 55 | |
| 101 | Lattice, Inc. | 114 | 365 | Team Austin, Inc. | 148 | |
| 118 | Lifeboat | 30 | 175 | TeleOperating Systems | 64 | |
| 366 | Logicware | 45 | 344 | Tom Rettig Association | 133 | |
| 257 | Logitech, Inc. | C-3 | 230 | True Basic | 119 | |
| 135 | Lugaru | 143 | * | The Software Family | 81 | |
| * | M & T Books & Software | * | 119 | * | Turbo Advantage Complex | 65 |
| 313 | Magma Systems | 85 | 332 | * | Turbo Advantage Display | 66 |
| 336 | Magus Inc. | 79 | 157 | * | Turbo Advantage Source Code | 67 |
| 108 | Manx Software Systems | 7 | 157 | * | Turbo Pascal Toolbook | 68 |
| 317 | Marshall Language Systems | 28 | 116 | Turbo Report | 57 | |
| 352 | Megamax | 97 | 116 | Unify Corporation | 115 | |
| 358 | MetaComCo | 137 | 116 | Vermont Creative Software | 50 | |
| 95 | MetaWare Incorporated | 107 | 116 | Vertical Horizons | 126 | |
| 300 | Micro Way | 135 | 116 | Wendin | 11 | |
| * | Micromint | 130 | 282 | Whitewater Group | 101 | |
| 84 | MicroPlot | 139 | 282 | Wizard Systems | 120 | |
| 154 | Microport Systems, Inc. | 23 | 244 | Workman & Associates | 105 | |
| 105 | Microprocessors Unlimited | 55 | 225 | Xenosoft | 55 | |
| 156 | Mix Software | 4-5 | | | | |
| 309 | Nanosoft Associates | 89 | | | | |

*This advertiser prefers to be contacted directly: see ad for phone number.

ADVERTISING SALES OFFICES

Midwest

Michelle Perkins (317) 875-8093

Southeast

Gary George (404) 897-1923

Northeast

Cynthia Zuck (718) 499-9333

Northern California/Northwest

Lisa Boudreau (415) 366-3600

Southern California/AZ/NM/TX

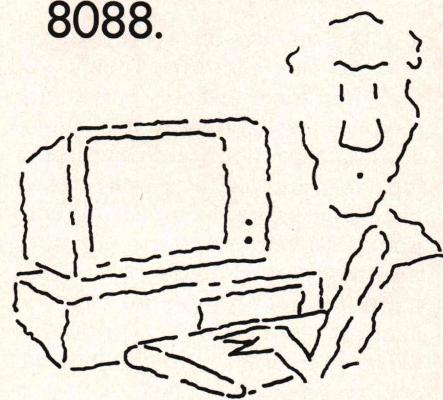
Michael Wiener (415) 366-3600

Advertising Director

Robert Horton (415) 366-3600

Get a Grip on Assembly Language.

The award winning
Visible Computer:
8088.



The Visible Computer is a book and software combination for mastering the elusive skills of assembly language. PC Tech Journal took one look and made it their September '85 "Program of the Month."

It's an animated simulation of the PC's microprocessor that lets you see with your own eyes how assembly language works. You'll be using it as a debugging tool for years to come.

It's a tutorial. A lot of people think the 350 page manual is the best book on assembly language ever written.

It's 45 demonstration programs you'll execute with the simulator, from simple register loads to advanced programs that manipulate interrupts and perform file I/O. And what you'll learn applies to all 86 family processors, including the \$79.95

80186 and 80286. not copy protected

The Visible Computer for IBM PC/XT/AT and true compatibles. If your dealer doesn't have it, order direct: Software Masters, 2714 Finfeather, Bryan, TX, 77801. (409) 822-9490. Please include \$3.00 shipping. Bank cards accepted.



TVC takes you inside the processor as it executes programs.

Software Masters™

Circle no. 347 on reader service card.

SWAINE'S FLAMES

"**Y**ou have discovered the truthism," Dennis Allison told me, "that there are only two things: memory and bandwidth." I had been arguing that, with memory becoming cheap and plentiful and *DDJ* having made its name squeezing computing power into scarce memory, the magazine should now concentrate on techniques for getting around bandwidth limitations. "Fine," he said, "but don't forget about memory."

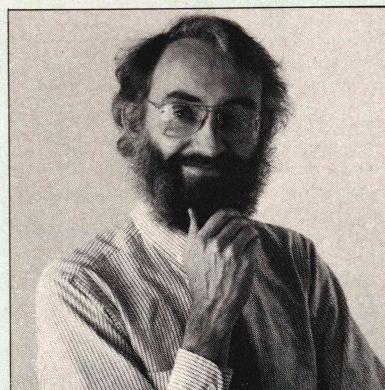
This column is about memory. I started writing it in a hotel room in Honolulu, a town in which Mark Twain, Jack London, and Hunter S. Thompson all wrote memorable reports on the Hawaiian islands; I had their books beside me as I turned on the television, looking for news of the Kilauea lava flow.

"The king . . . could place a taboo upon any spot or thing or person and it was death for any man to molest it."—*The Sandwich Islands*, Mark Twain.

Remember GNU, the outrageous nonproprietary Unix-like operating system-in-progress? Richard Stallman says he is about to start the kernel. The C compiler is now cranking out 68020 code.

Richard enjoys defying taboos.

Levi Thomas represented *DDJ* at the second Hacker's Conference, a gathering made memorable by the spectacle of Ted Nelson, Jerry Pournelle, and Timothy Leary on the same stage and by the reminiscences of Chris Espinosa and the brothers Baum. In the midst of it all Jolt arrived and was promptly declared programming fuel. Jolt is a soft drink that boasts "all the sugar and twice the caffeine"—sounds like something you wouldn't want to drink just before applying for a job at IBM, FMC,



Syntex, Lockheed, or any of the other companies now requiring drug testing of job applicants.

And let's not forget the West Coast Computer Faire. The Advisory Committee for next month's Faire (which includes Lee Felsenstein and *DDJ*'s co-founder Dennis Allison, first editor Jim Warren, and current assistant editor Levi Thomas) is concerned that the Faire has lost much of the excitement that in the past has made it more than a trade show (and, I suspect, wants to prevent WCCF from becoming the unanimous disappointment that the PC Faire has become). The committee has plans for the Faire that it hopes will rekindle some of that remembered excitement. One notion: Chinese-menu-style hot-pepper symbols next to advanced topics in the program to warn novices. Also: a keynote panel on who really owns the IBM standard, a pioneers' panel on computers and the receding vision of utopia, and some early reports from the field on 386 implementations. And then there's Jerry Pournelle's "I'm mad as hell and I'm not going to take it any more" session.

Those who attended the December 17, 1986, meeting of the Homebrew Computer Club at Stanford University have something to remember. The club's perennial toastmaster, Lee Felsenstein, opened the gathering for the last time that night. (Homebrew is where Processor Technology [remember it?] got its first orders for S-100 memory boards and where Steve Wozniak showed off his Apple I.)

And they'll have something to remember it by—cartoonist Larry Gonick produced a commemorative T-shirt for the event.

After the meeting, the attendees retired to their traditional watering hole, The Oasis, for peanuts, beer, and wood carving.

I remember they fed us peanuts on the flight to the Kona Coast, where I hoped to get as close as I could get to Kilauea.

"Kona, where nobody ever dreams of looking at a thermometer, where every afternoon there falls a refreshing spring shower, and where neither frost nor sunstroke has ever been known . . . a hurricane . . . a fog . . . are meteorologically impossible . . ."—*My Hawaiian Aloha*, Jack London.

"The Kona Coast in December is as close to hell on earth as a half-bright mammal can get."—*The Curse of Lono*, Hunter S. Thompson.

Both London's and Thompson's memories were faulty. I have seen the debris of a terrible hurricane on a Hawaiian beach and beautiful weather on the Kona Coast in December. And, both terrible and beautiful, the plume of steam rising from the sea where molten lava was building more Hawaii. I suspect I'll remember that when much that seems urgent now is forgotten.

My cousin Corbett's last words as he got on the plane for Hawaii this morning were, "When the Earth gets weird, the weird go into real estate."

A handwritten signature of Michael Swaine in black ink.

Michael Swaine
editor-in-chief

LOGITECH MODULA-2/86 HOLIDAY PACKAGE

\$89 Price

- Separate Compilation
- Native Code Generation
- Large Memory Model Support
- Multitasking
- Powerful Debugging Tools
- Comprehensive Module Library
- Available for the PC and the VAX

Use LOGITECH MODULA-2/86 to decrease your overall development cycle and produce more reliable, more maintainable code.

 **LOGITECH
MODULA-2/86** **\$89**

Includes Editor, Run Time System, Linker, 8087 Software Emulation, Binary Coded Decimal (BCD) Module, Logitech's comprehensive library, Utility to generate standard .EXE files. AND more!

 **LOGITECH MODULA-2/86
with 8087 Support** **\$129**

 **LOGITECH MODULA-2/86
PLUS** **\$189**

For machines with 512K of RAM.
Increases compilation speed by 50%.

 **RUN TIME DEBUGGER
(Source level!)** **\$69**

The ultimate professional's tool! Display source, data, call chain and raw memory. Set break points, variables, pinpoint bugs in your source!

 **UTILITIES PACKAGE** **\$49**

Features a Post-Mortem Debugger (PMD). If your program crashes at run-time the PMD allows you to analyze the status of the program and locate the error. Also includes a Disassembler, Cross Reference Utility, and Version that allows conditional compilation.

 **LIBRARY SOURCES** **\$99**

Source code now available for customization or exemplification.



WINDOW PACKAGE **\$49**

Build windows into your programs. Features virtual screens, color support, overlapping windows and a variety of borders.

MAKE UTILITY **\$29**

Figures out dependencies and automatically selects modules affected by code changes to minimize recompilation and relinking.

CROSS RUN TIME **\$199** **Debugger and ROM Package**

Still available at an introductory price!

TURBO PASCAL to MODULA-2 TRANSLATOR **\$49**

"Turbo Pascal... is a very good system. But don't make the mistake of trying to use it for large programs."

*Niklaus Wirth**

Our Translator makes it even easier for Turbo users to step up to Modula-2/86. It changes your Turbo source code into Modula-2/86 source, solves all the incompatibilities, and translates the function calls of Turbo into Modula-2/86 procedures. Implements the complete Turbo libraries!

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

30 Day Money Back Guarantee!

To place an order call our special toll free number:

800-231-7717

in California

800-552-8885

\$199

Special
Holiday Offer

Step up to the power of LOGITECH MODULA-2/86 at a saving of nearly \$100 off our usual low prices! We're offering a complete tool set including our compiler with 8087 support (for use with or without an 8087), our Turbo to Modula-2/86 Translator, Run Time Debugger, and Utilities in one holiday package at a special price!

YES I want to step up to LOGITECH MODULA-2/86!

Here's the configuration I'd like:

| | | |
|--------------------------|----------------------------|-------|
| <input type="checkbox"/> | Special Holiday Package | \$199 |
| <input type="checkbox"/> | Logitech Modula-2/86 | \$89 |
| <input type="checkbox"/> | with 8087 support | \$129 |
| <input type="checkbox"/> | Plus Package | \$189 |
| <input type="checkbox"/> | Turbo to Modula Translator | \$49 |
| <input type="checkbox"/> | Run Time Debugger | \$69 |
| <input type="checkbox"/> | Utilities Package | \$49 |
| <input type="checkbox"/> | Library Sources | \$99 |
| <input type="checkbox"/> | Window Package | \$49 |
| <input type="checkbox"/> | Make Utility | \$29 |
| <input type="checkbox"/> | ROM Package | \$199 |

Add \$6.50 for shipping and handling. Calif. residents add applicable sales tax. Prices valid in U.S. only.

Total Enclosed \$_____

Visa MasterCard Check Enclosed

Card Number Expiration Date

Signature

Name

Address

City State Zip

Phone



LOGITECH

Logitech, Inc.
805 Veterans Blvd.
Redwood City, CA 94063
Tel: 415-365-9852

In Europe:
Logitech SA, Switzerland
Tel: 41-21-879656

In Italy: Tel: 39-2-215-5622

New Turbo Prolog and Turbo Pascal Toolboxes add more Power

New! Turbo Prolog Toolbox

Our new Turbo Prolog Toolbox™ enhances Turbo Prolog—with more than 80 tools and over 8,000 lines of source code that can easily be incorporated into your programs. It includes about 40 example programs that show you how to use your new tools.

New Turbo Prolog Toolbox features include:

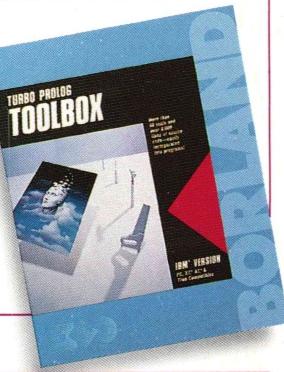
- Business graphic generation
- Complete communications package
- File transfers from Reflex, dBASE III, 1-2-3, Symphony
- A unique parser generator
- Sophisticated user-interface design tools

System requirements

Turbo Prolog: IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K. Turbo Prolog Toolbox requires Turbo Prolog 1.10 or higher. Dual-floppy disk drive or hard disk. 512K.

NEW

Only
\$99.95



- Matrix operations: inversions, determinants and eigenvalues
- Least squares approximations
- Differential equations

As well as a free demo FFT program, you also get Least Squares Fit in 5 different forms.

1. Power
2. Exponential
3. Logarithm
4. 5-term Fourier
5. 5-term Polynomial

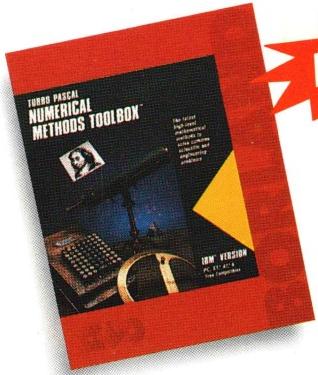
They're all ready to compile and run.

All this for only \$99.95

NEW

Only
\$99.95

The power and high performance of Turbo Pascal® is already in the hands of more than half-a-million people. The technically superior Turbo Pascal is the de facto worldwide standard and the clear leader.



New! Turbo Pascal Numerical Methods Toolbox

What our new Numerical Methods Toolbox™ will do for you now:

- Find solutions to equations
- Interpolations
- Calculus: numerical derivatives and integrals
- Fourier transforms

System requirements
IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. Turbo Pascal 2.0 or later. Graphics module required. Monitor with IBM CGA, IBM EGA, or Hercules compatible adapter card, and requires Turbo Graphix Toolbox. 8087 or 80287 numeric co-processor not required, but recommended for optimal performance. 256K.

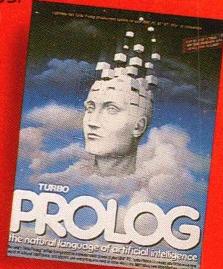
Turbo Pascal 3.0:
Includes 8087 & BCD features for 16-bit MS-DOS and CP/M-86 systems. CP/M-80 version minimum memory: 48K; 8087 and BCD features not available. 128K.

Turbo Prolog, the natural language of Artificial Intelligence

Turbo Prolog™ is the most popular AI package in the world with more than 100,000 users. It's the 5th-generation computer programming language that brings supercomputer power to your IBM® PC and compatibles. You can join the AI revolution with Turbo Prolog for only \$99.95.

Step-by-step tutorials, demo programs and source code included.

Only
\$99.95!



"If you're at all interested in artificial intelligence, databases, expert systems, or new ways of thinking about programming, by all means plunk down your \$100 and buy a copy of Turbo Prolog.

Bruce Webster, BYTE '89

For the dealer nearest you, or to order by phone
call (800) 255-8008

CA (800) 742-1133
Canada (800) 237-1136



Turbo Graphix Toolbox, Turbo Pascal, and Reflex are registered trademarks and Turbo Prolog, Turbo Prolog Toolbox, and Turbo Pascal Numerical Methods Toolbox are trademarks of Borland International, Inc. or Borland Analytica, Inc. dBASE III is a registered trademark of Ashton-Tate. Lotus 1-2-3 and Symphony are registered trademarks of Lotus Development Corp. IBM, XT, and AT are registered trademarks of International Business Machines Corp. Hercules is a trademark of Hercules Computer Technology. CP/M is a registered trademark of Digital Research, Inc. MS-DOS is a registered trademark of Microsoft Corp. Copyright 1986 Borland International

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX 172373